Incremental Data Flow analysis using PRISM

Rashmi Rekha Mech
(*Project Guide: Prof. Uday Khedker*)

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

March 2015

Outline of the talk

- Incremental Data Flow analysis
- Requirements
  - Present Architectures
  - Required Architecture
- Incremental Analysis in details

Part I

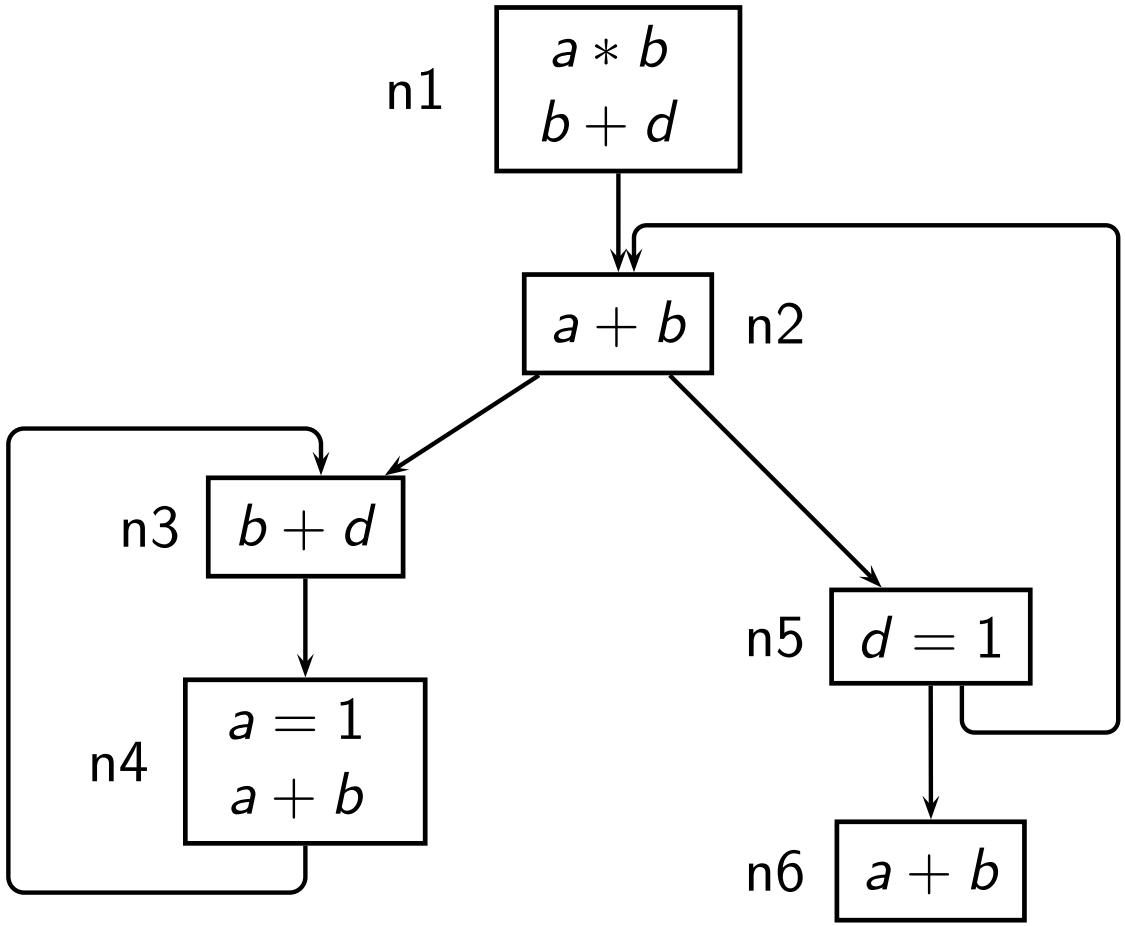
Incremental Data Flow analysis

Why Incremental Analysis?

When program undergoes changes:

- Some or all computed data flow information become invalid
- Recomputation is required

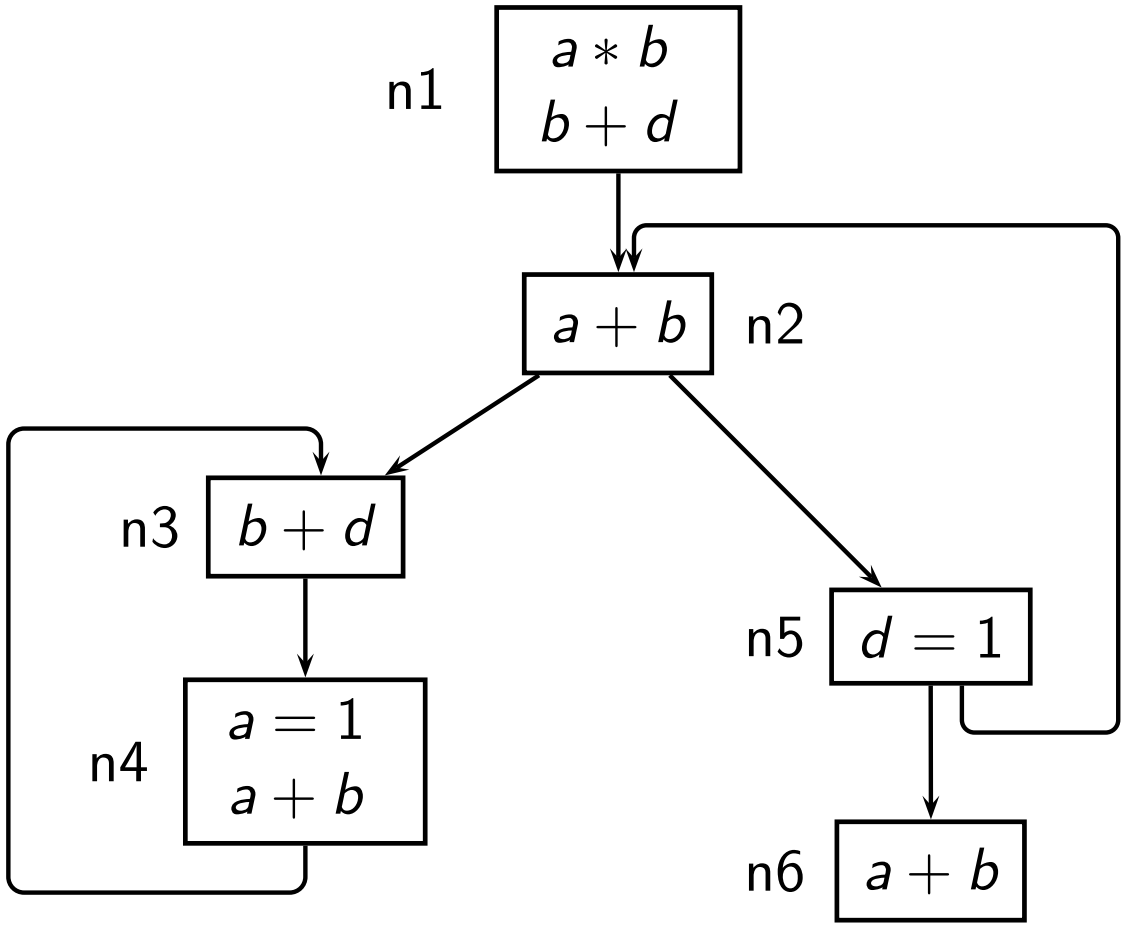
Motivating Example - Available Expression Analysis



Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis

1st Iteration

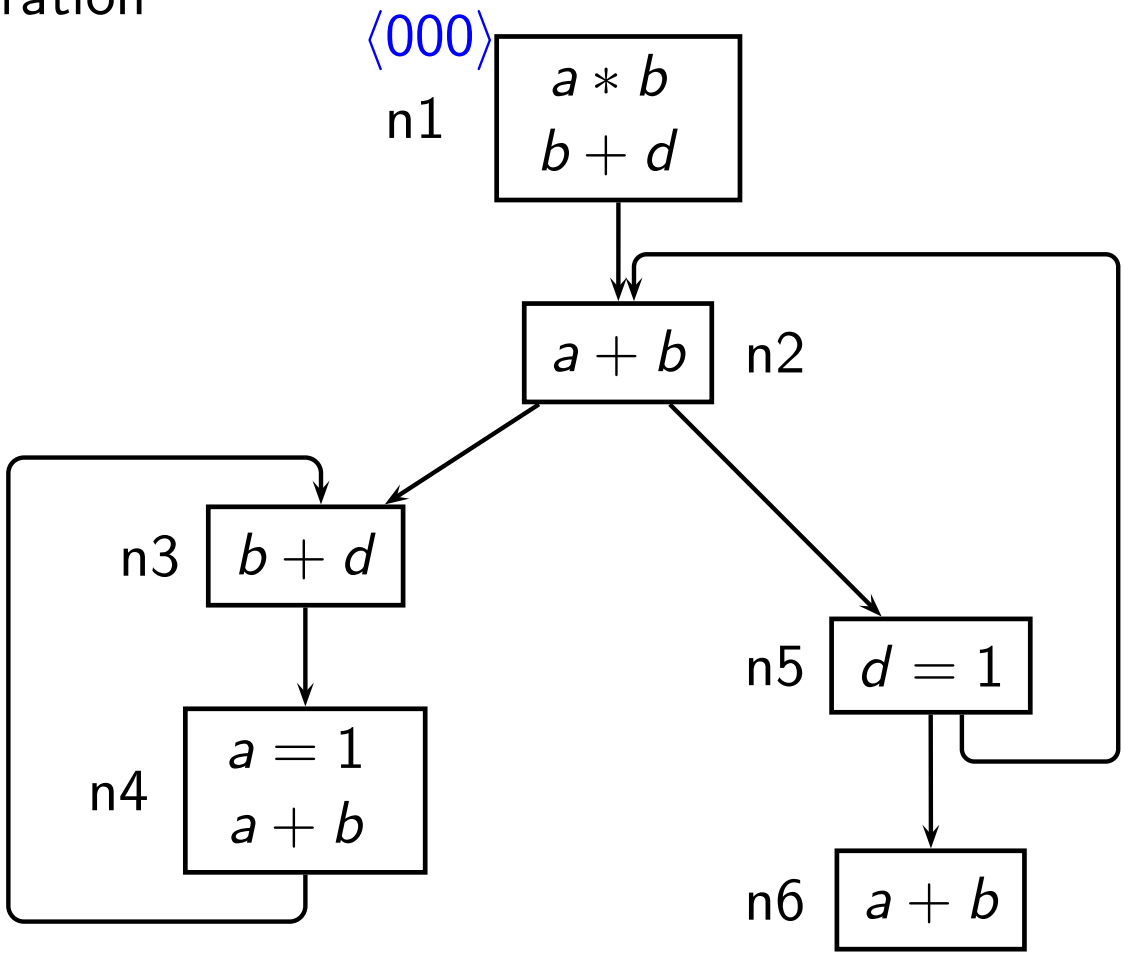


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

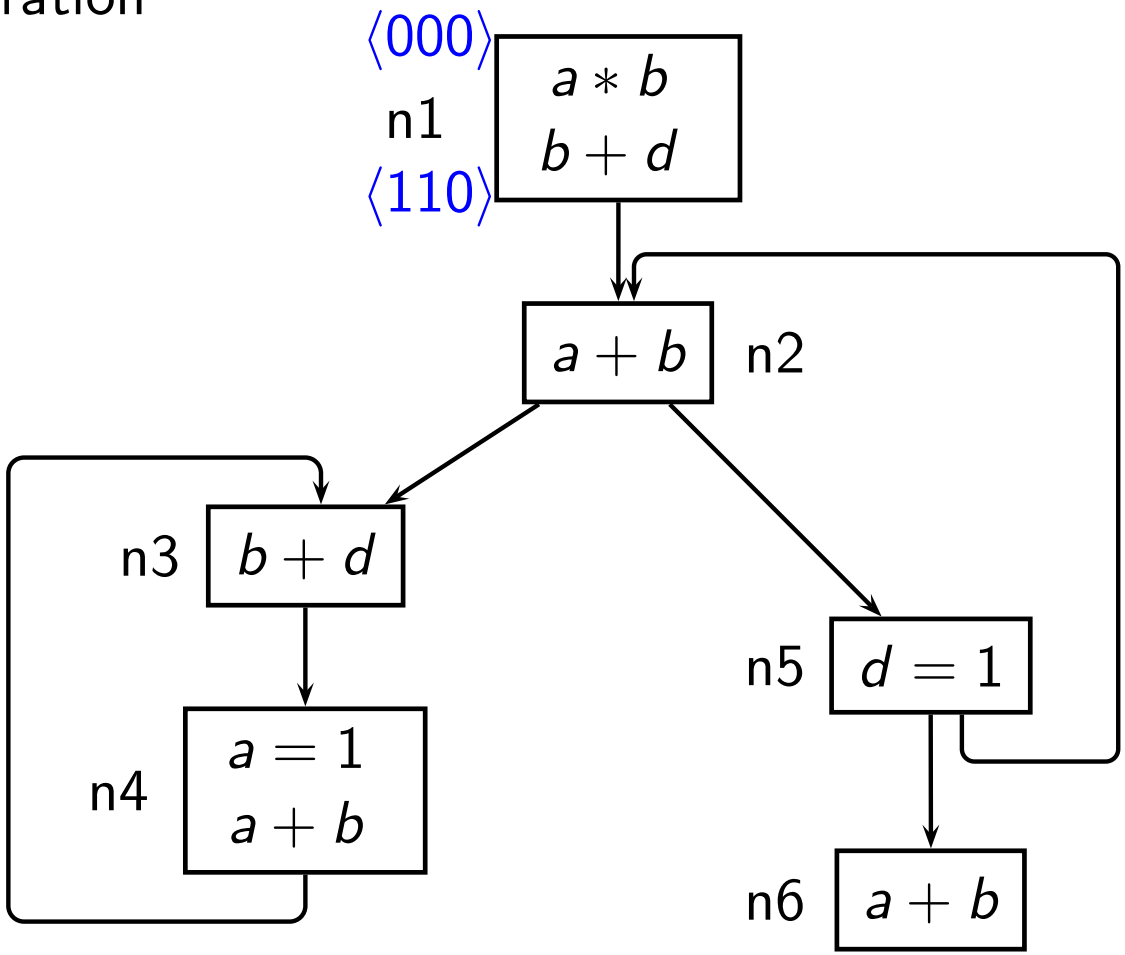
1st Iteration



Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis

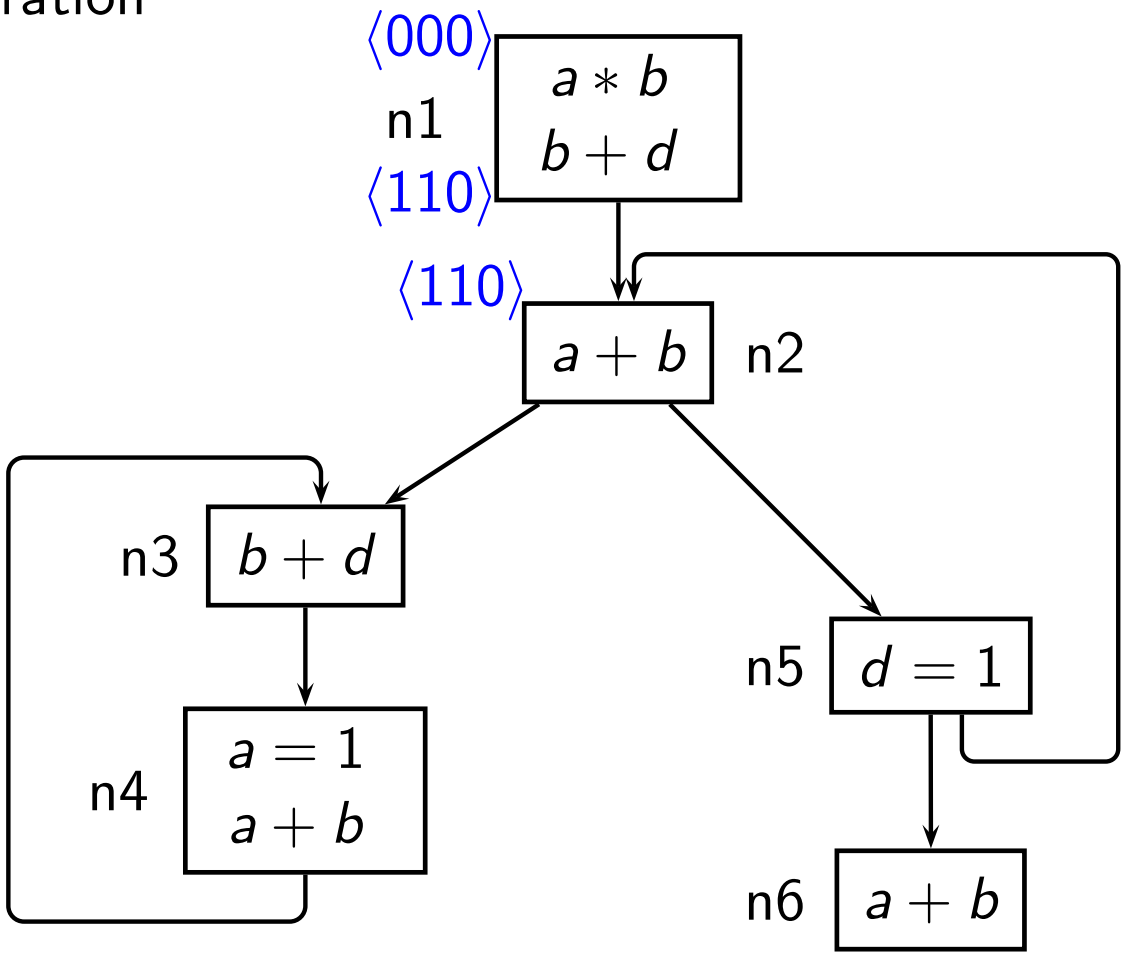
1st Iteration



Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis

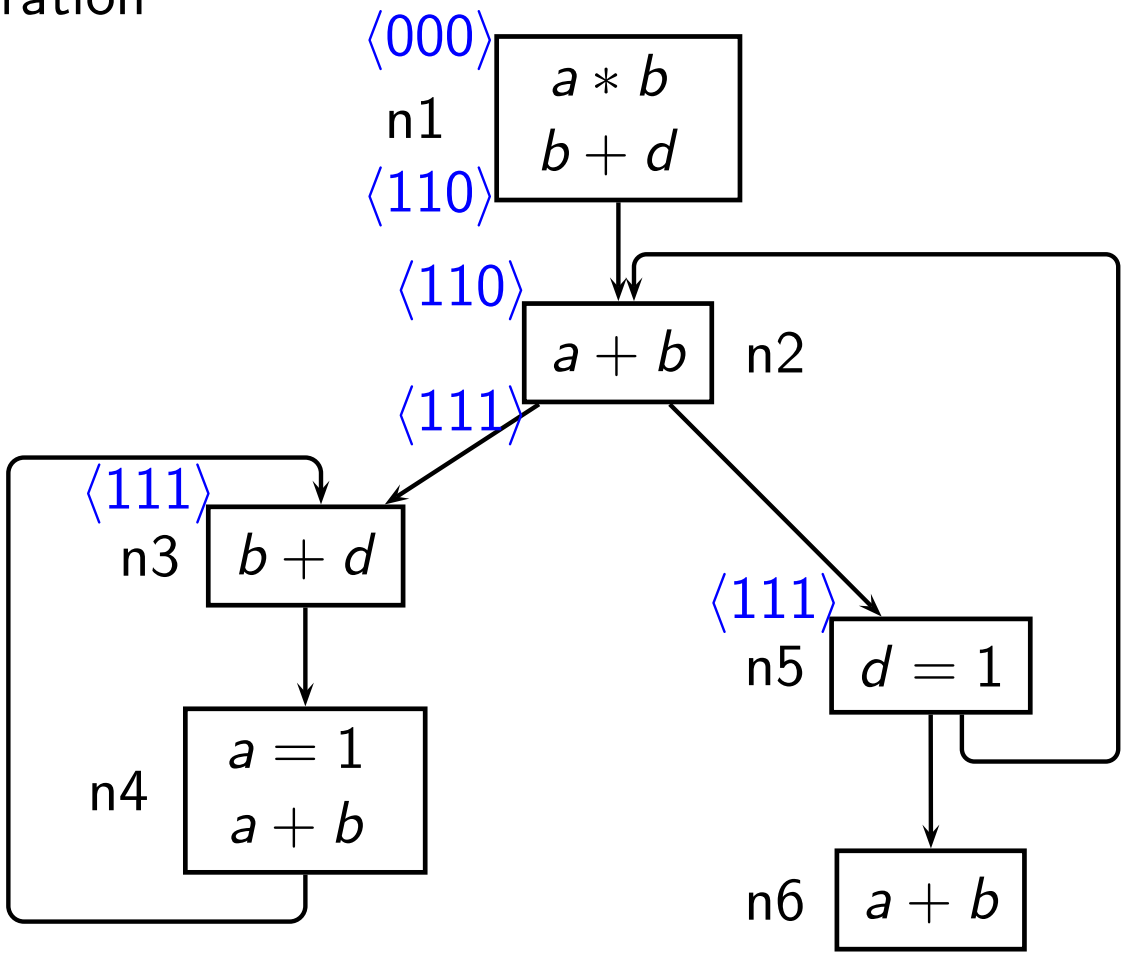
1st Iteration



Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis

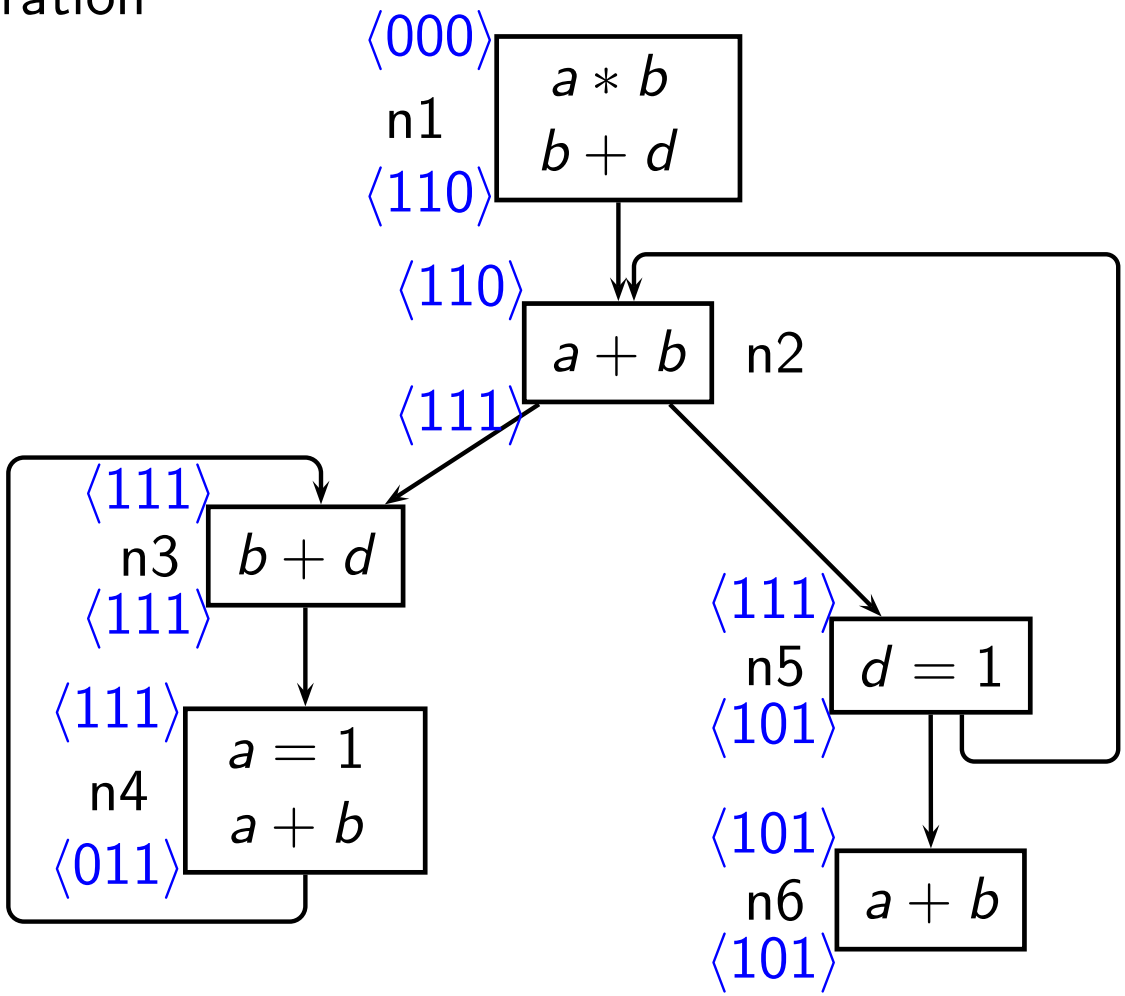
1st Iteration



Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis

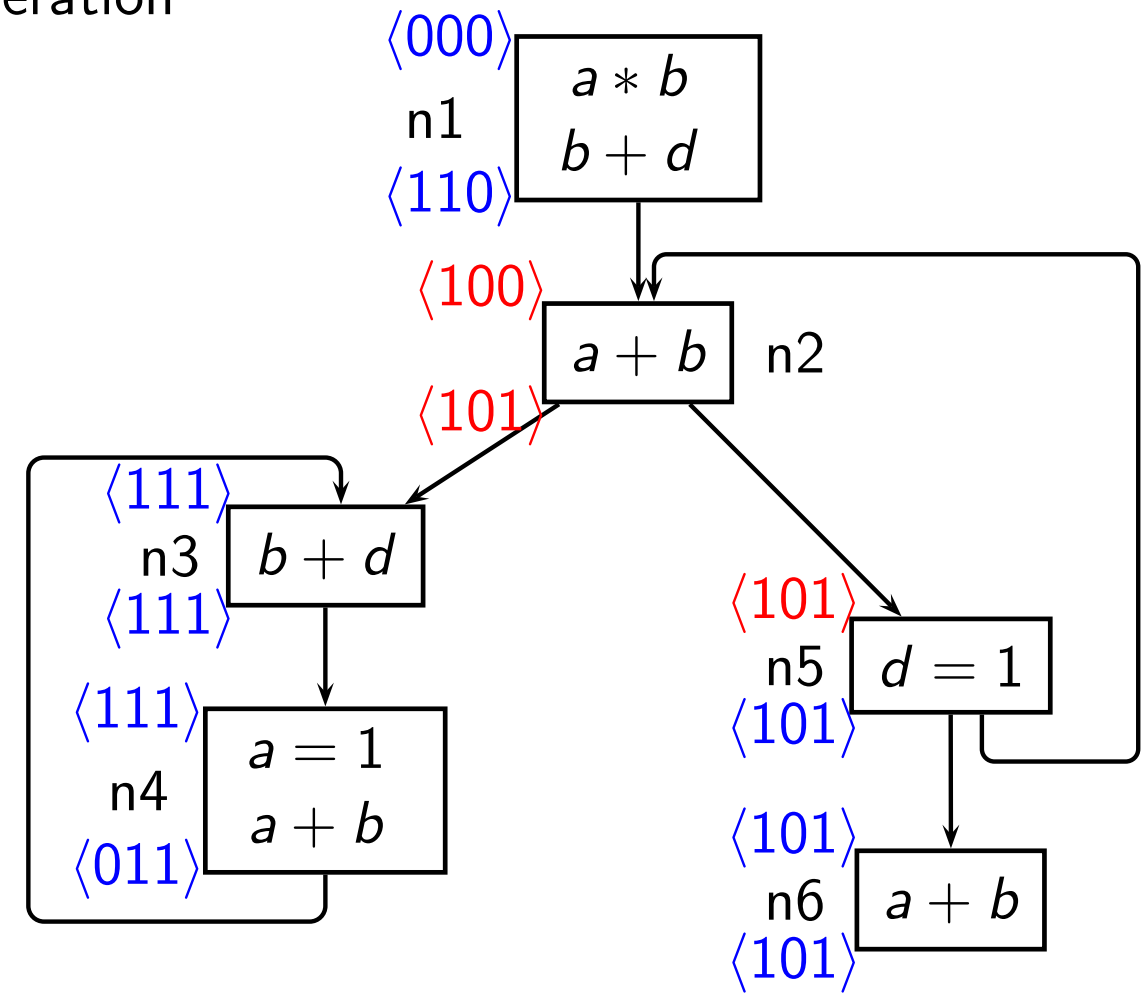
1st Iteration



Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis

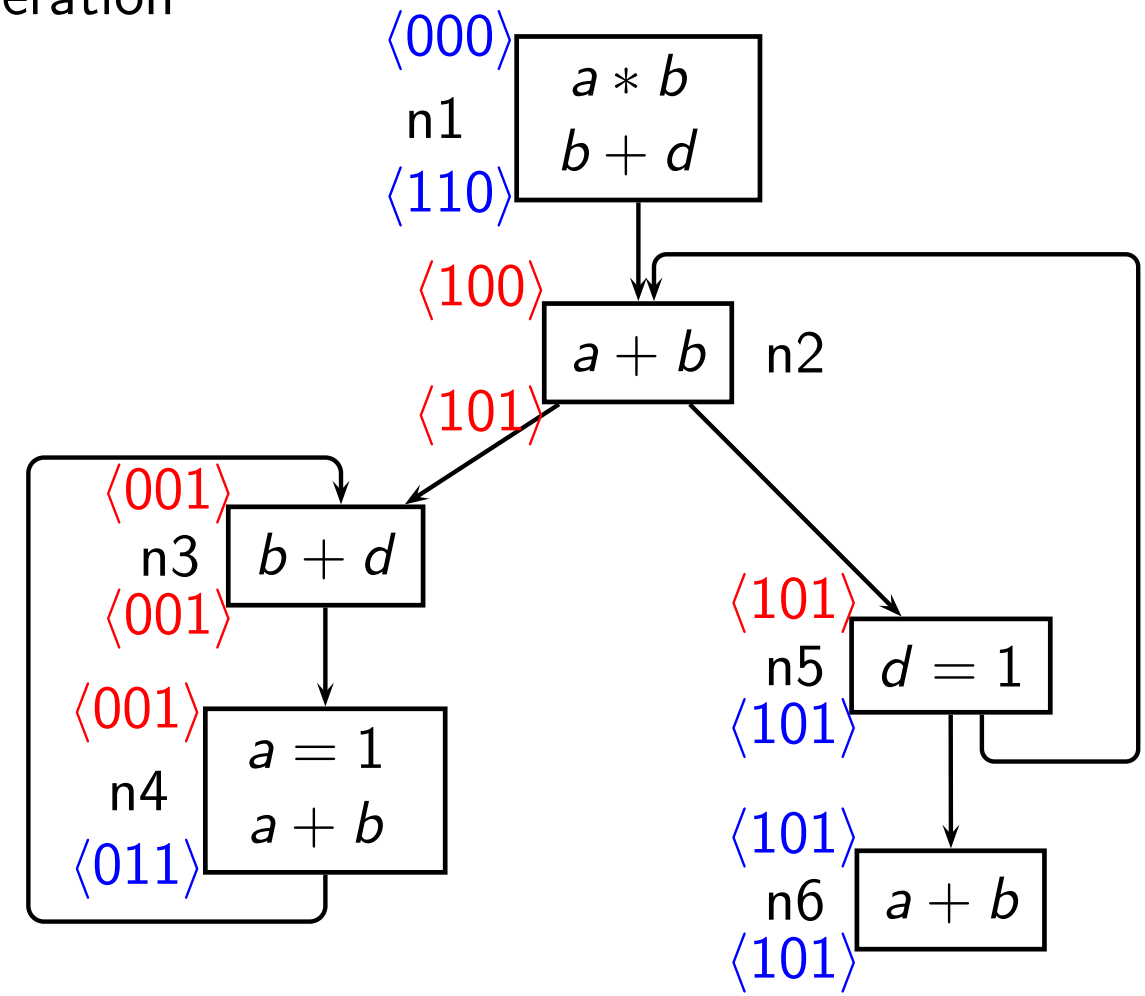
2nd Iteration



Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis

2nd Iteration

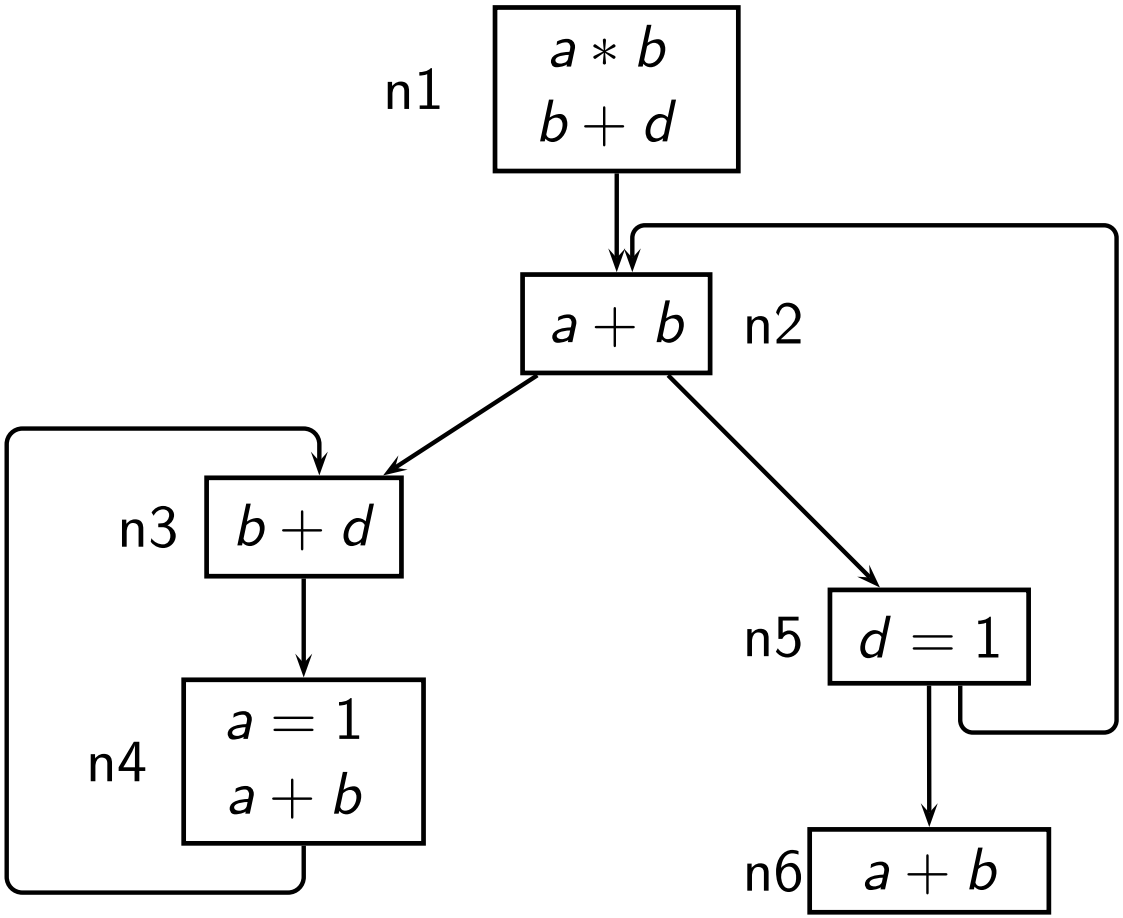


Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis

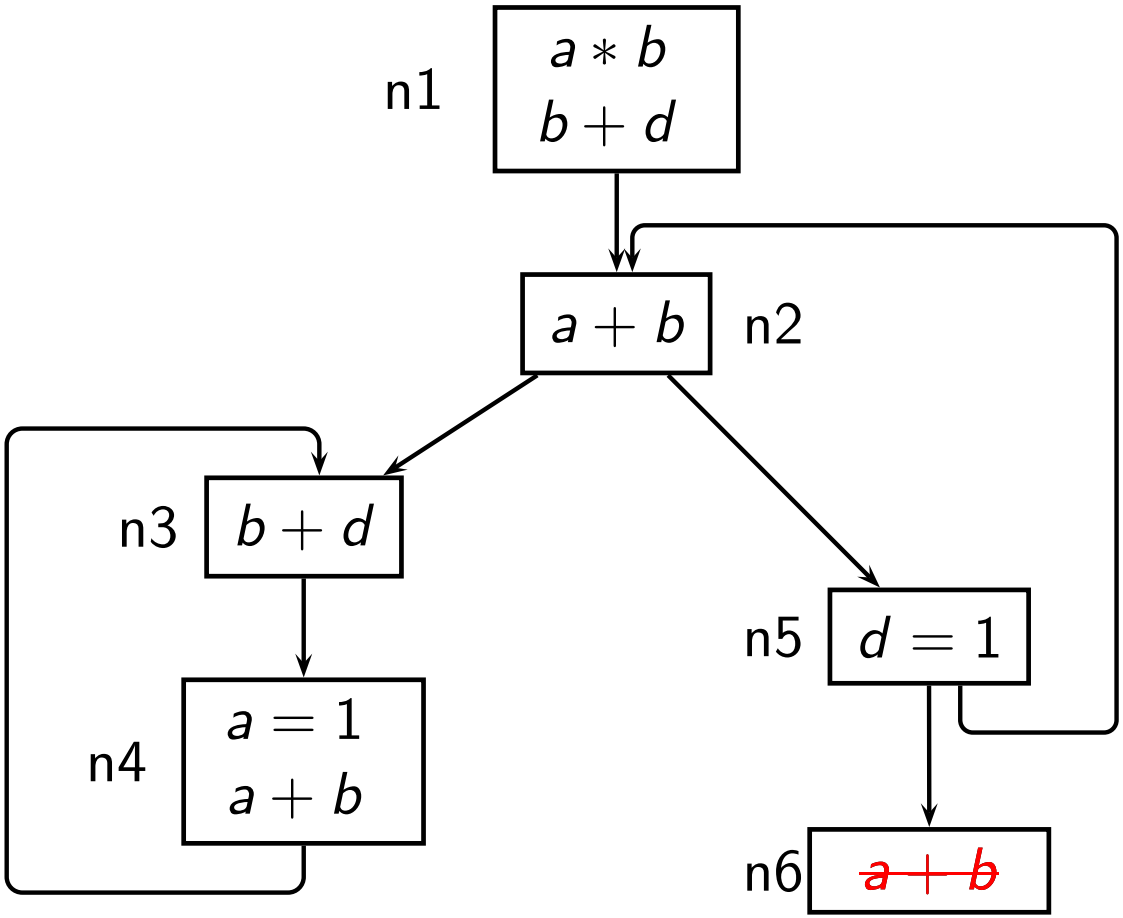
- It requires 3 iterations to converge

Motivating Example - Available Expression Analysis



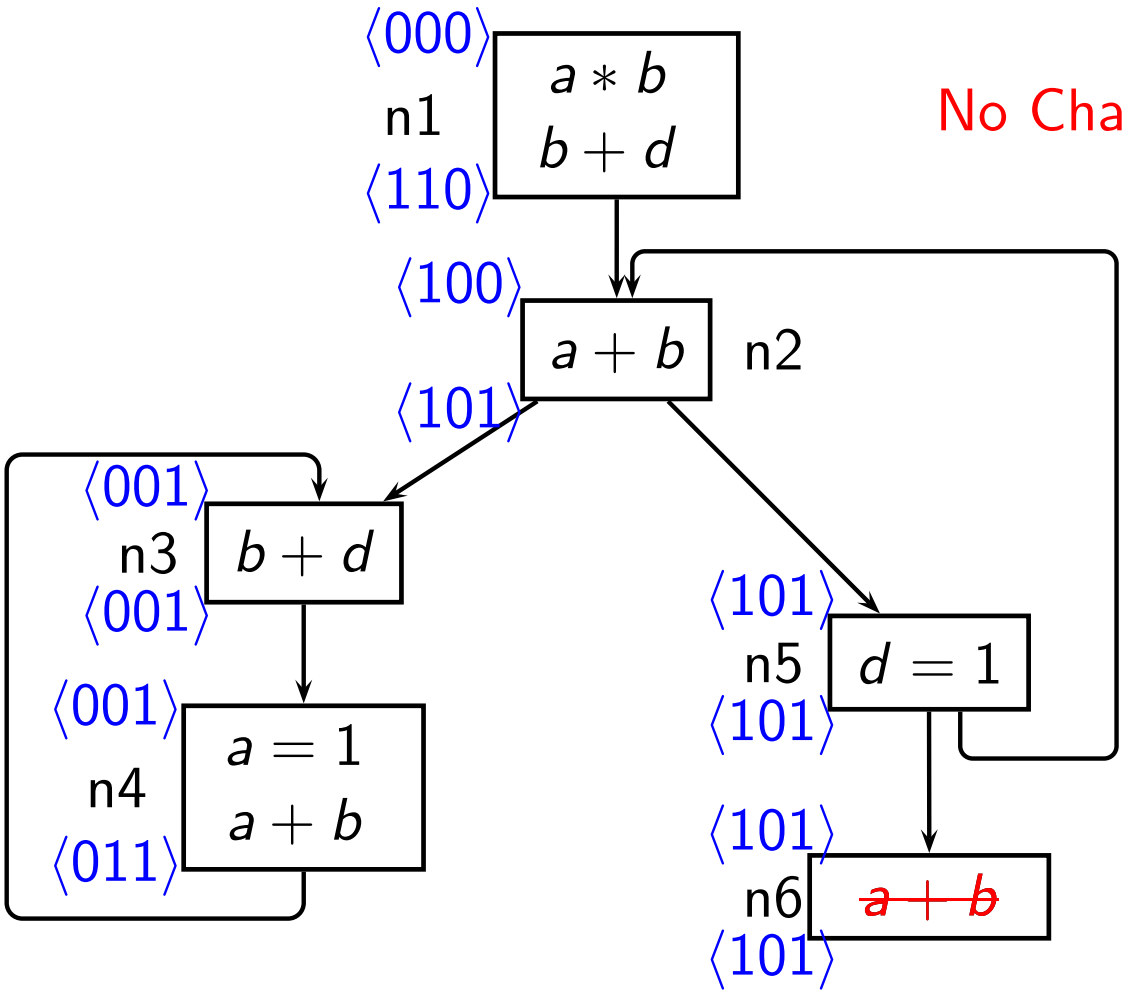
Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis



Bit Vector | $a * b$ | $b + d$ | $a + b$ |

Motivating Example - Available Expression Analysis



No Change in data flow values

Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

- Recomputing the values from the scratch is very inefficient
- Need an incremental analysis:
  - modifies only affected data flow information
  - more cost effective then **exhaustive** analysis in terms of space and time

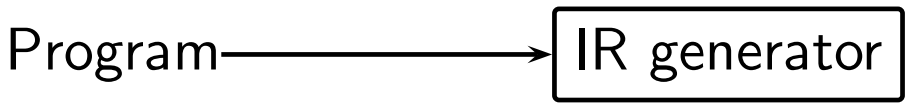
Part II

Requirements

Present Architecture of PRISM

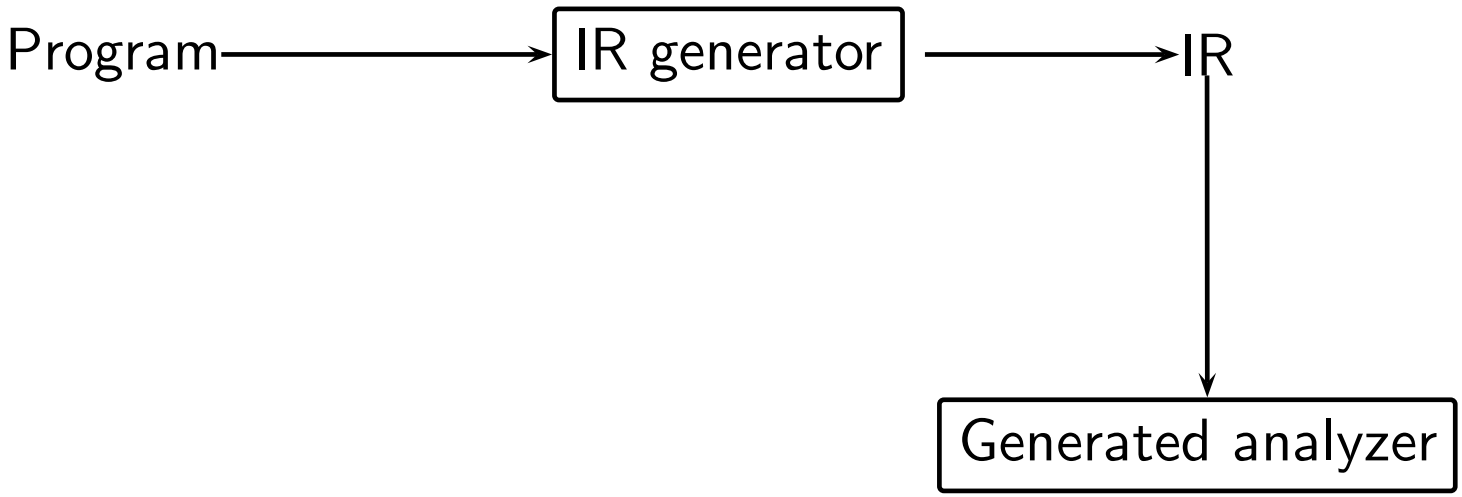
Program

Present Architecture of PRISM

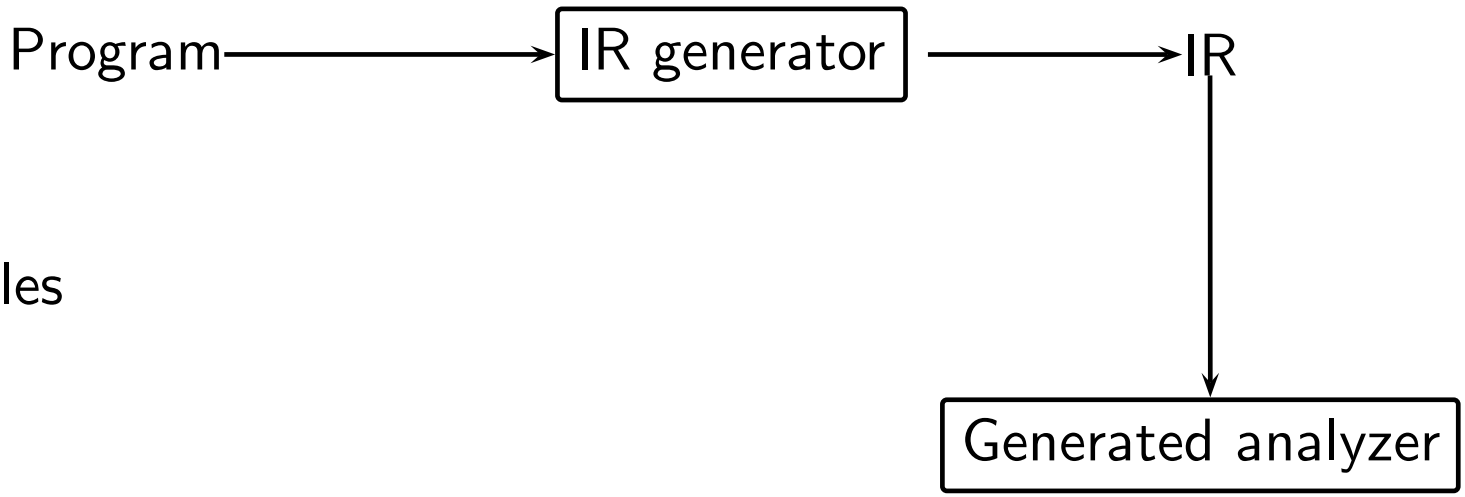


Present Architecture of PRISM

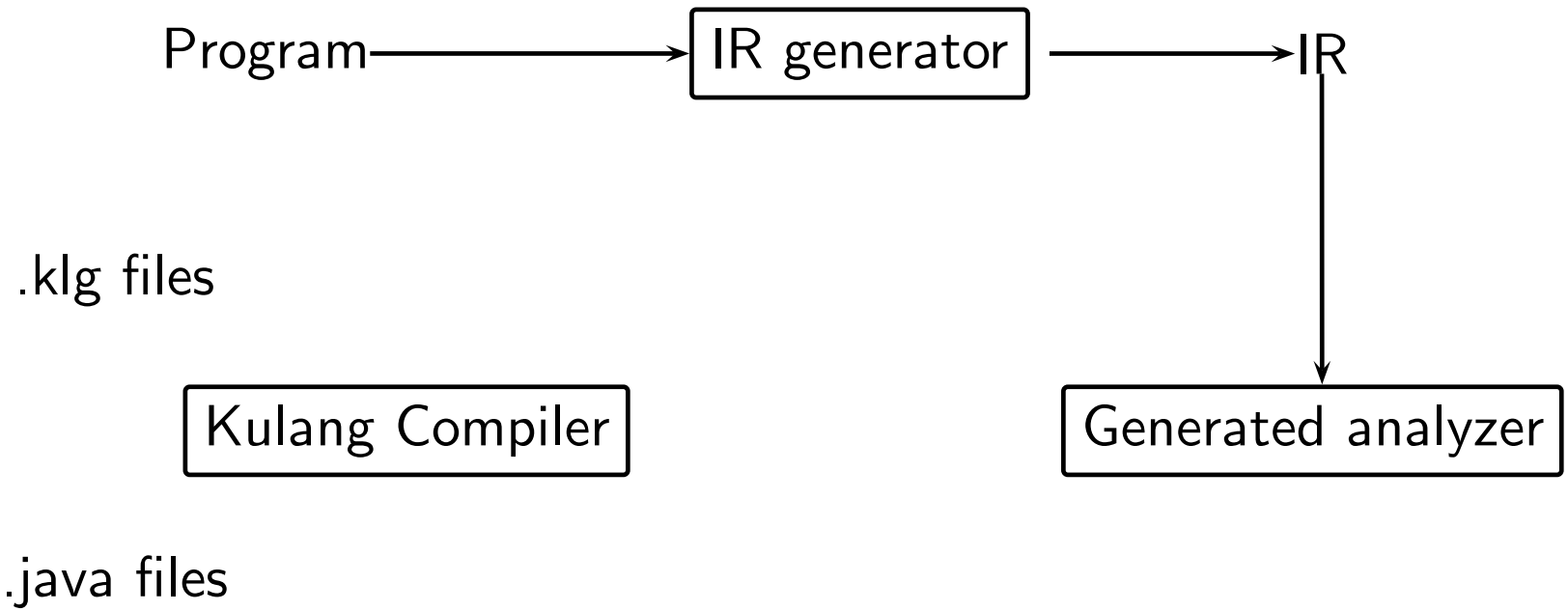
Present Architecture of PRISM



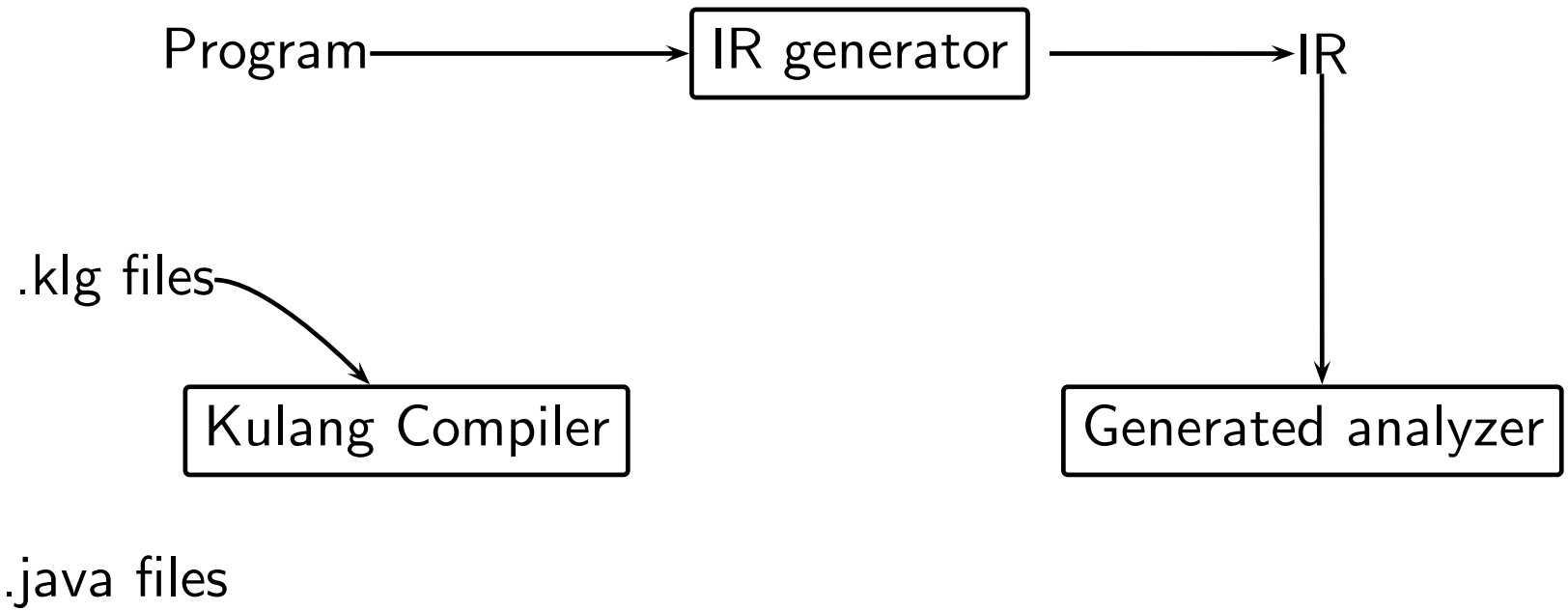
Present Architecture of PRISM



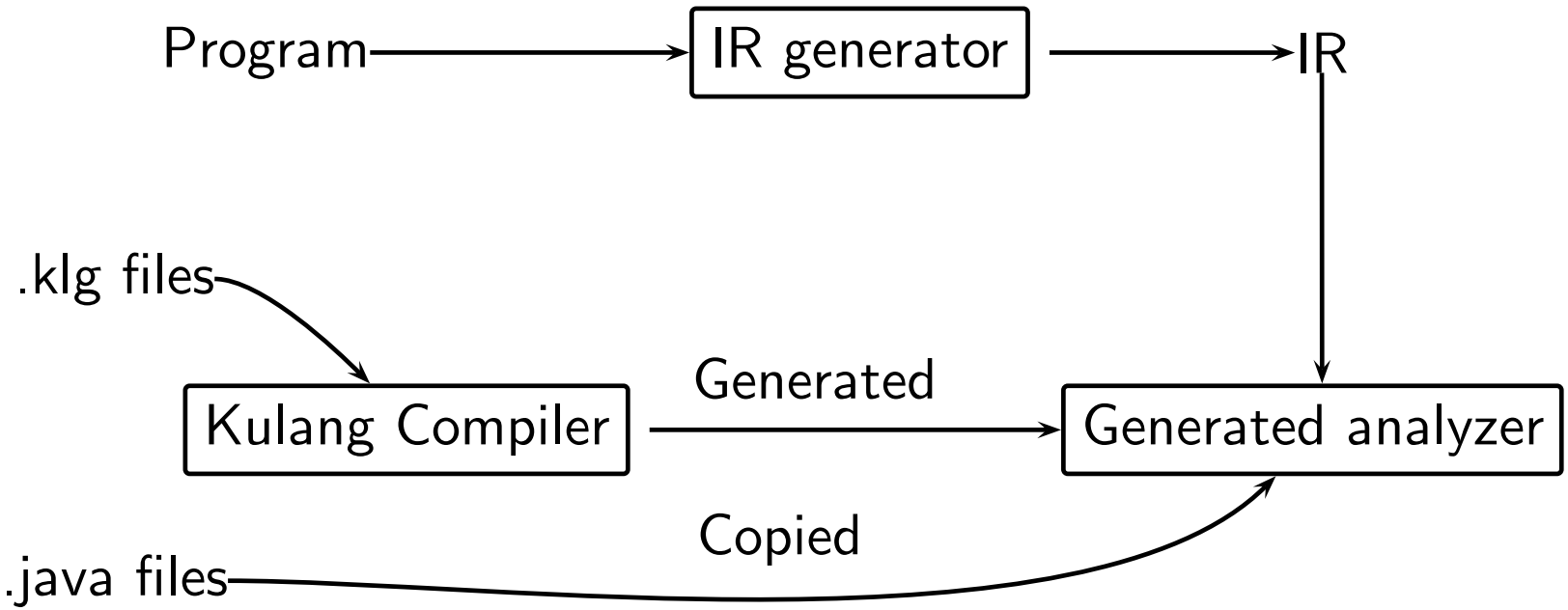
Present Architecture of PRISM



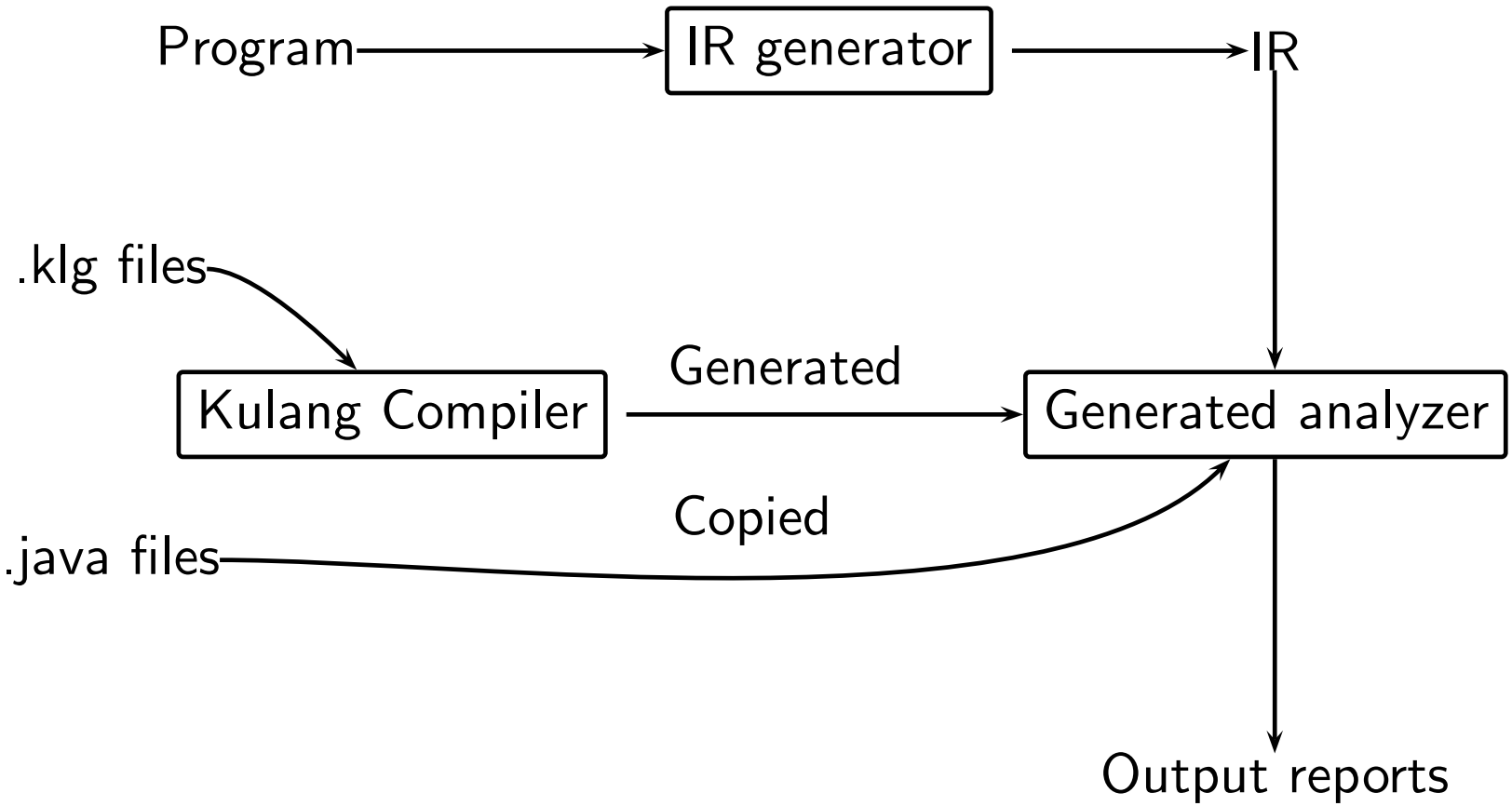
Present Architecture of PRISM



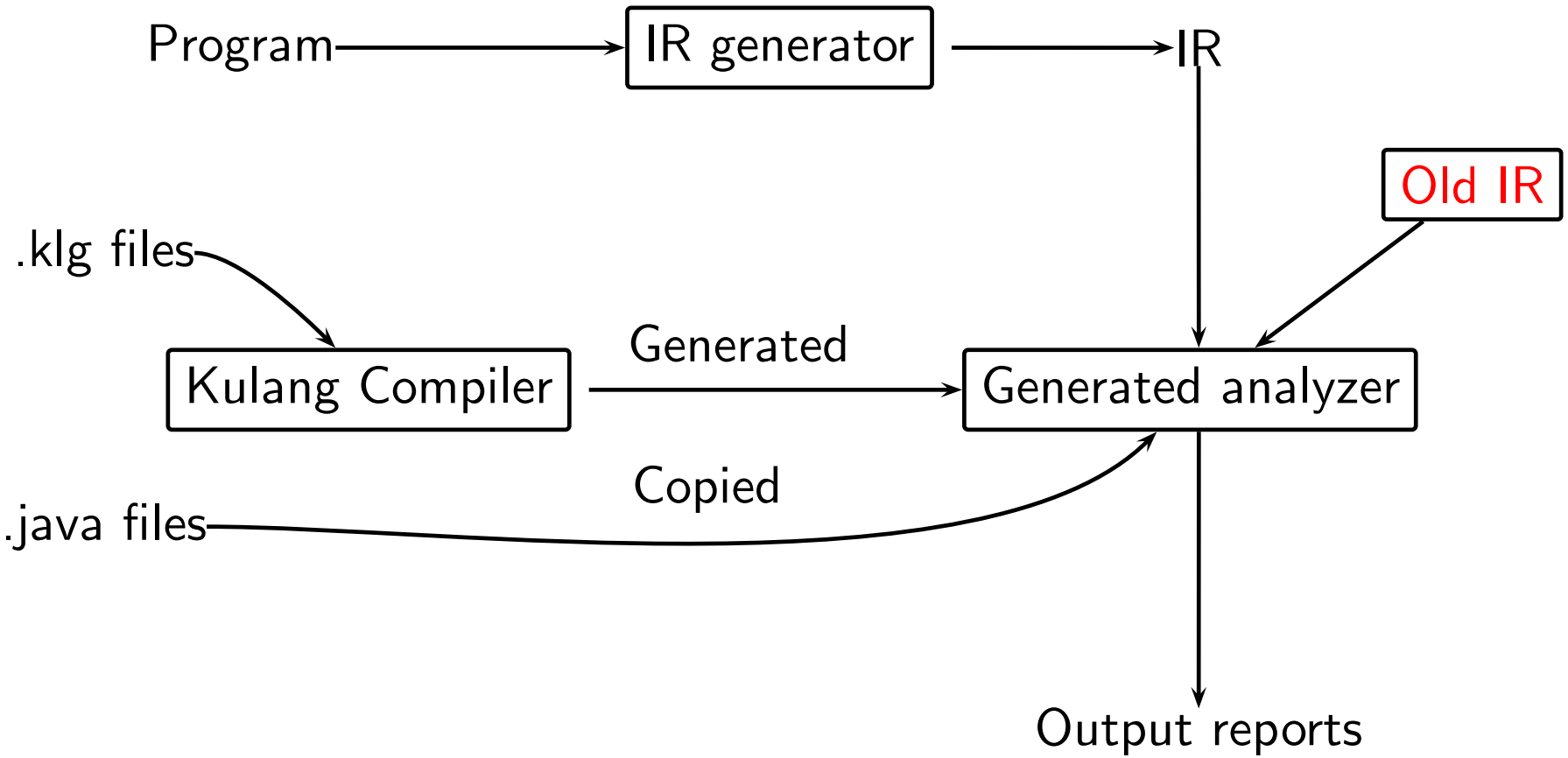
Present Architecture of PRISM



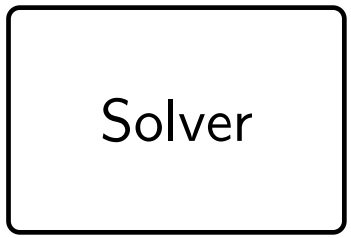
Present Architecture of PRISM



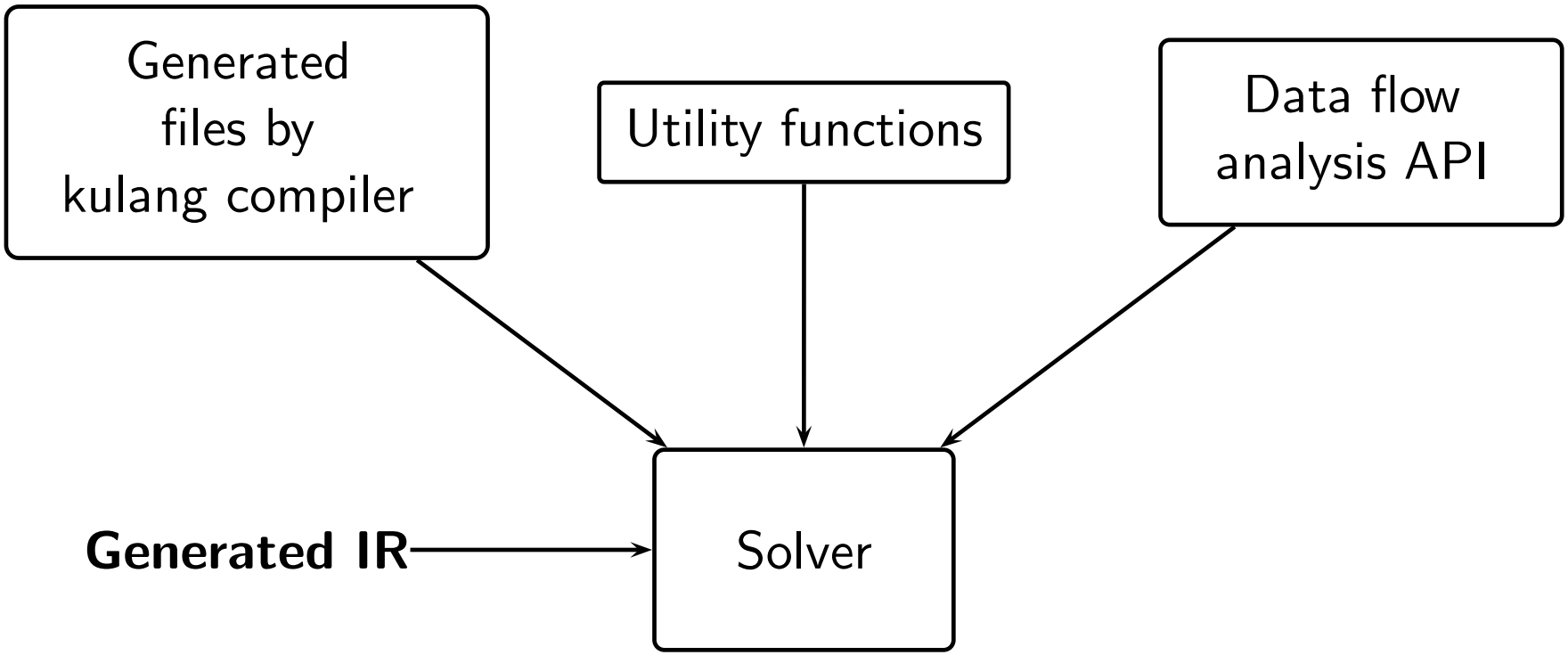
Present Architecture of PRISM



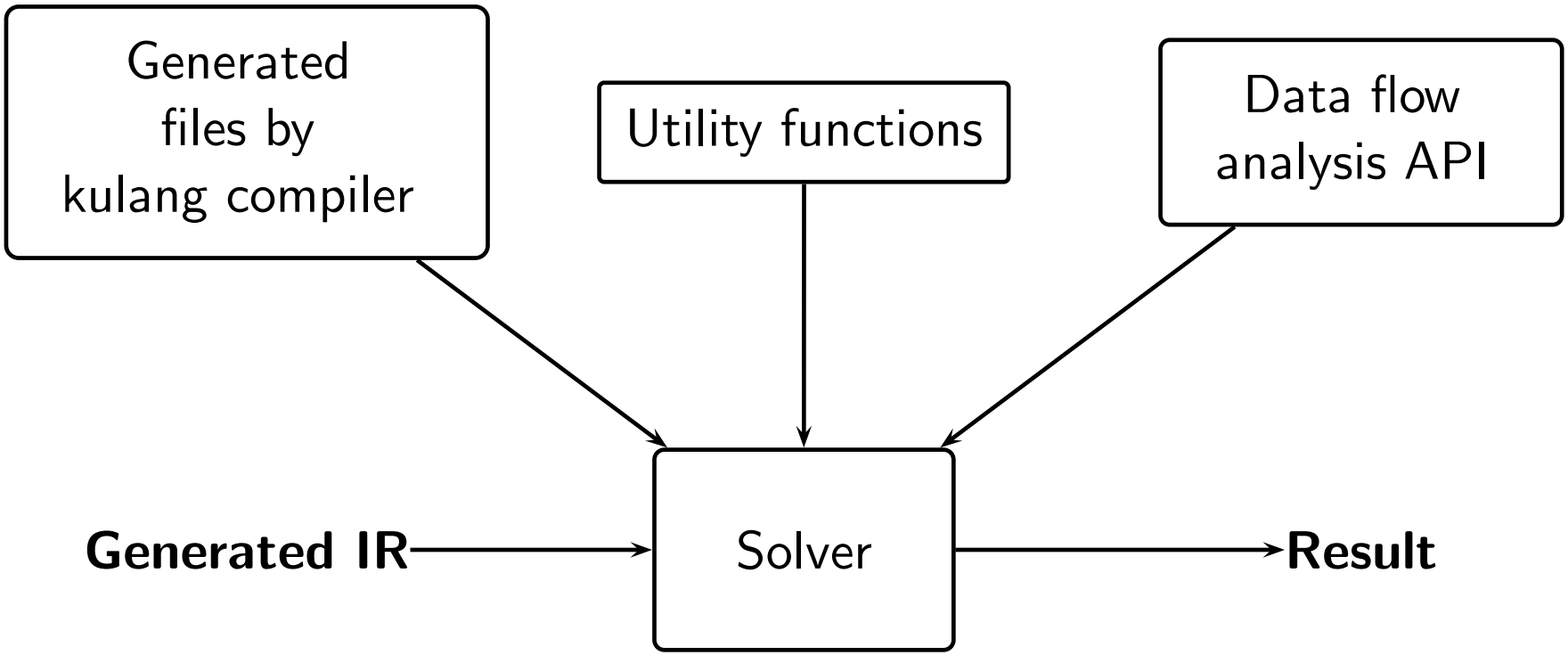
Present architecture of analyser generator



Present architecture of analyser generator



Present architecture of analyser generator



Required architecture of analyser generator

Generated IR

old IR

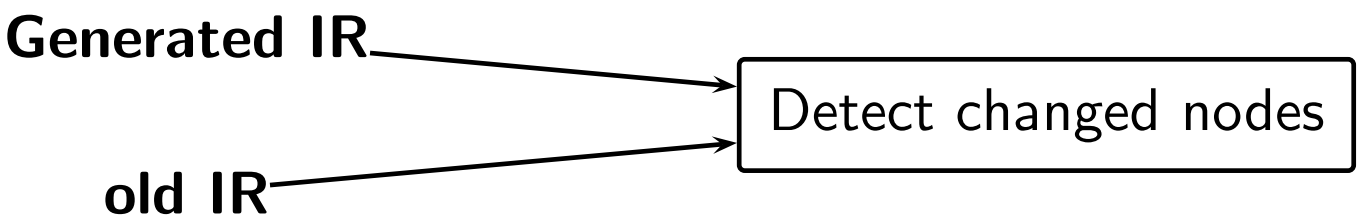
Required architecture of analyser generator

Generated IR

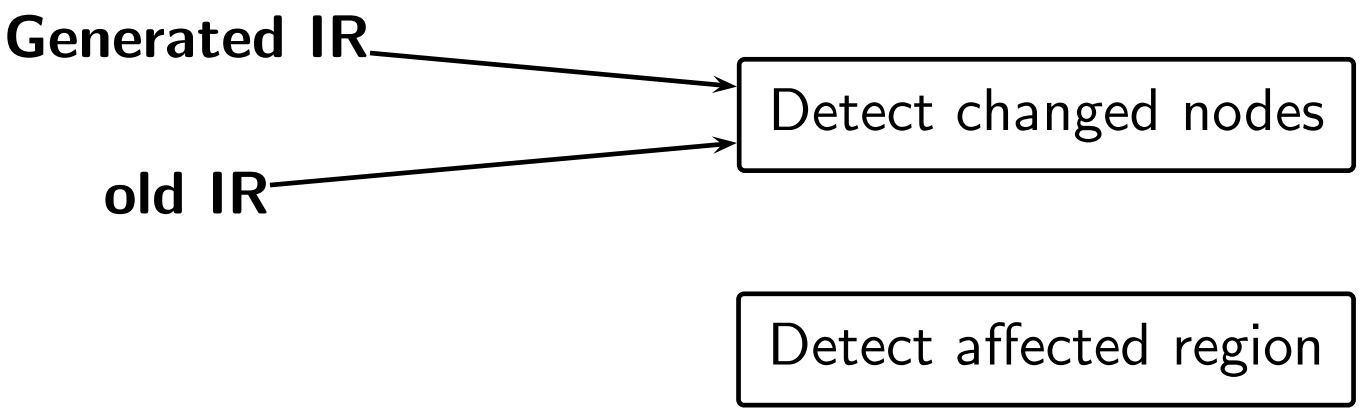
Detect changed nodes

old IR

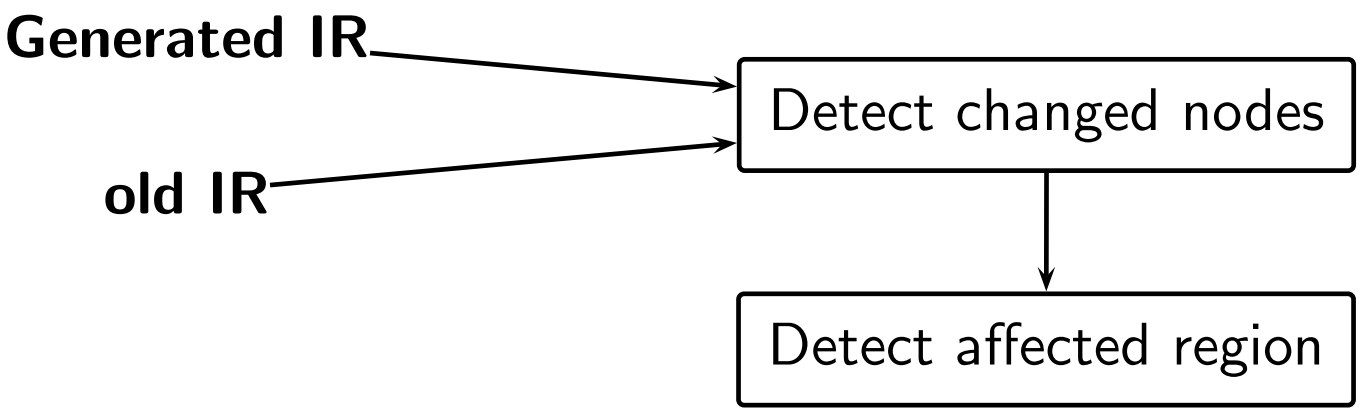
Required architecture of analyser generator



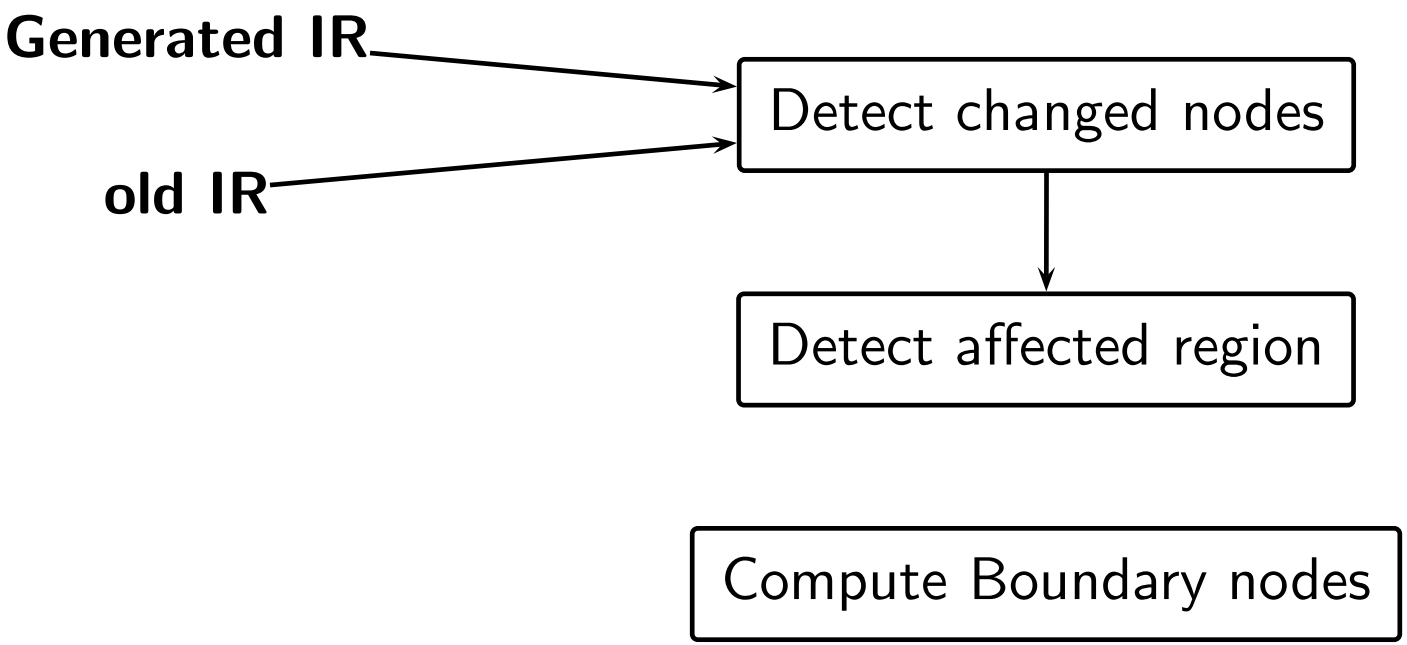
Required architecture of analyser generator



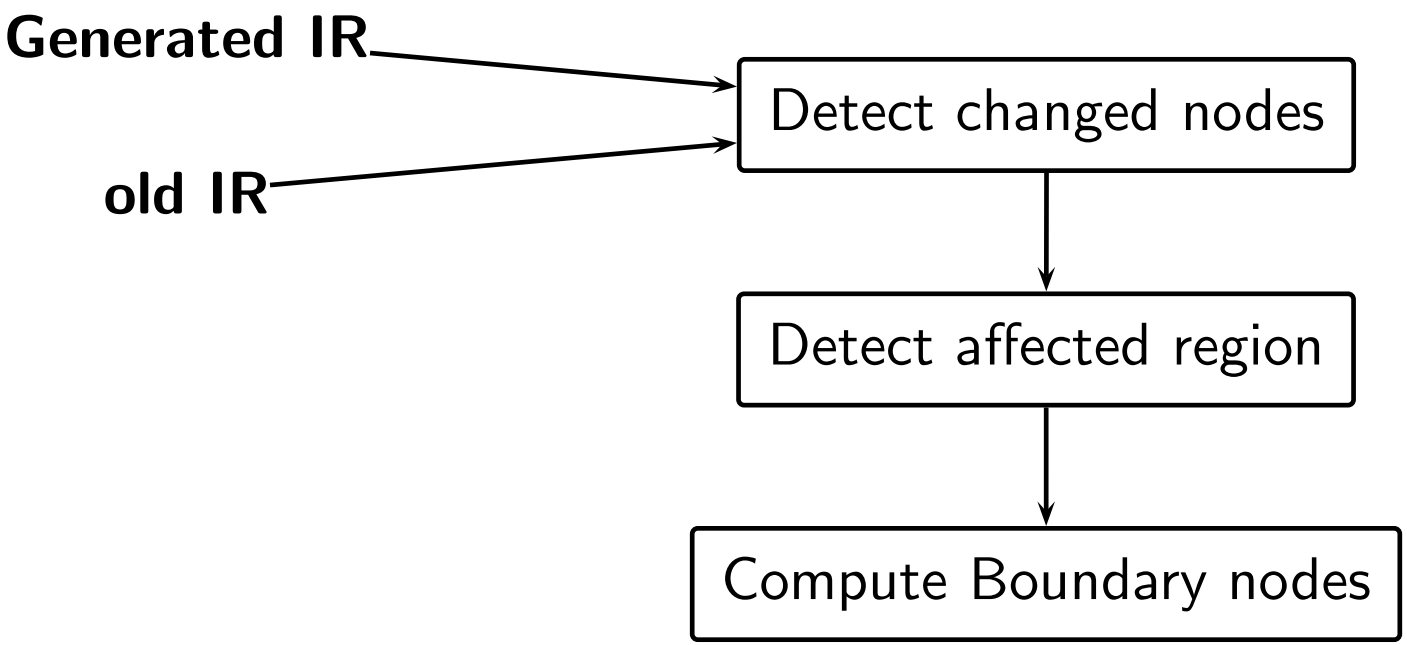
Required architecture of analyser generator



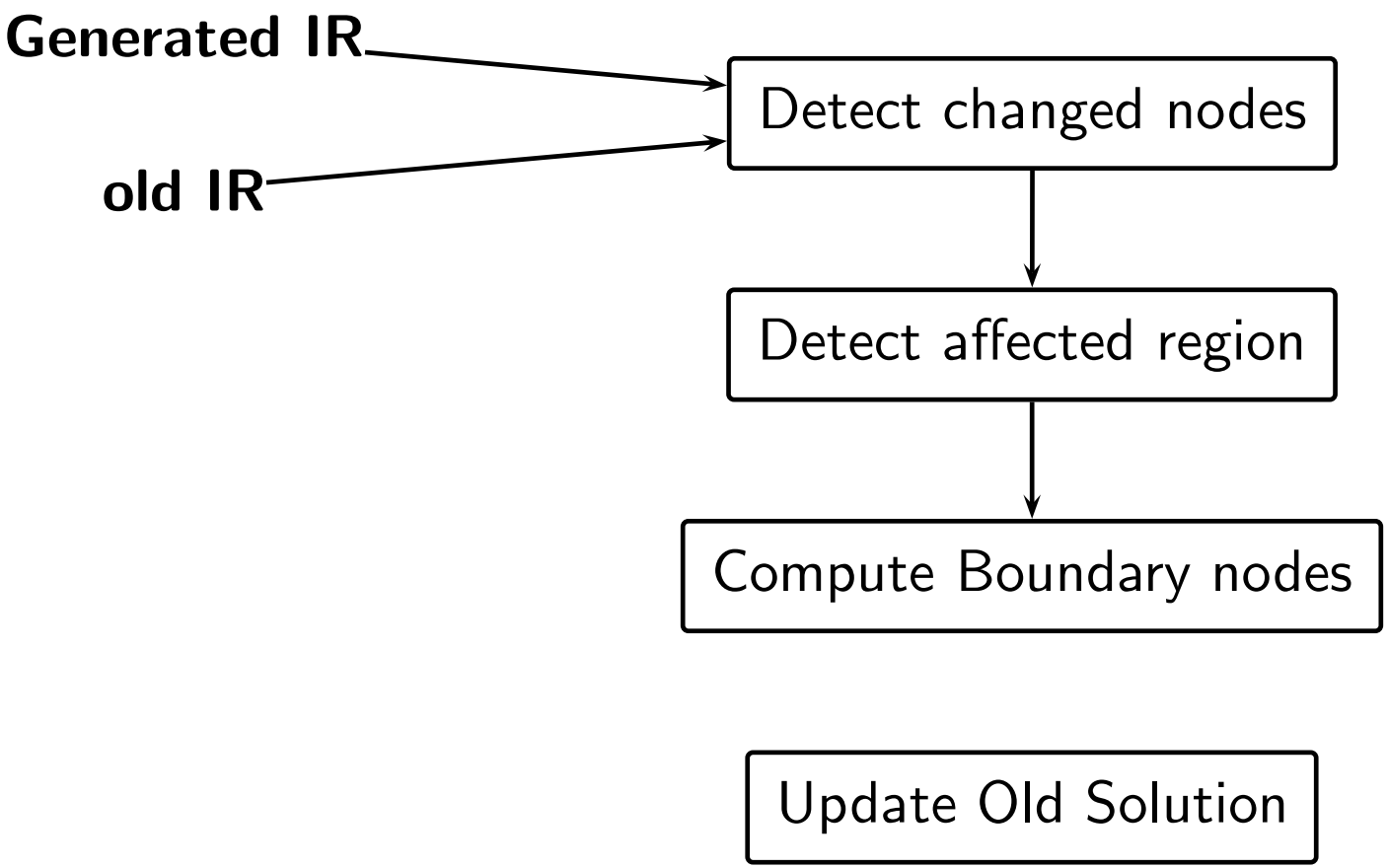
Required architecture of analyser generator



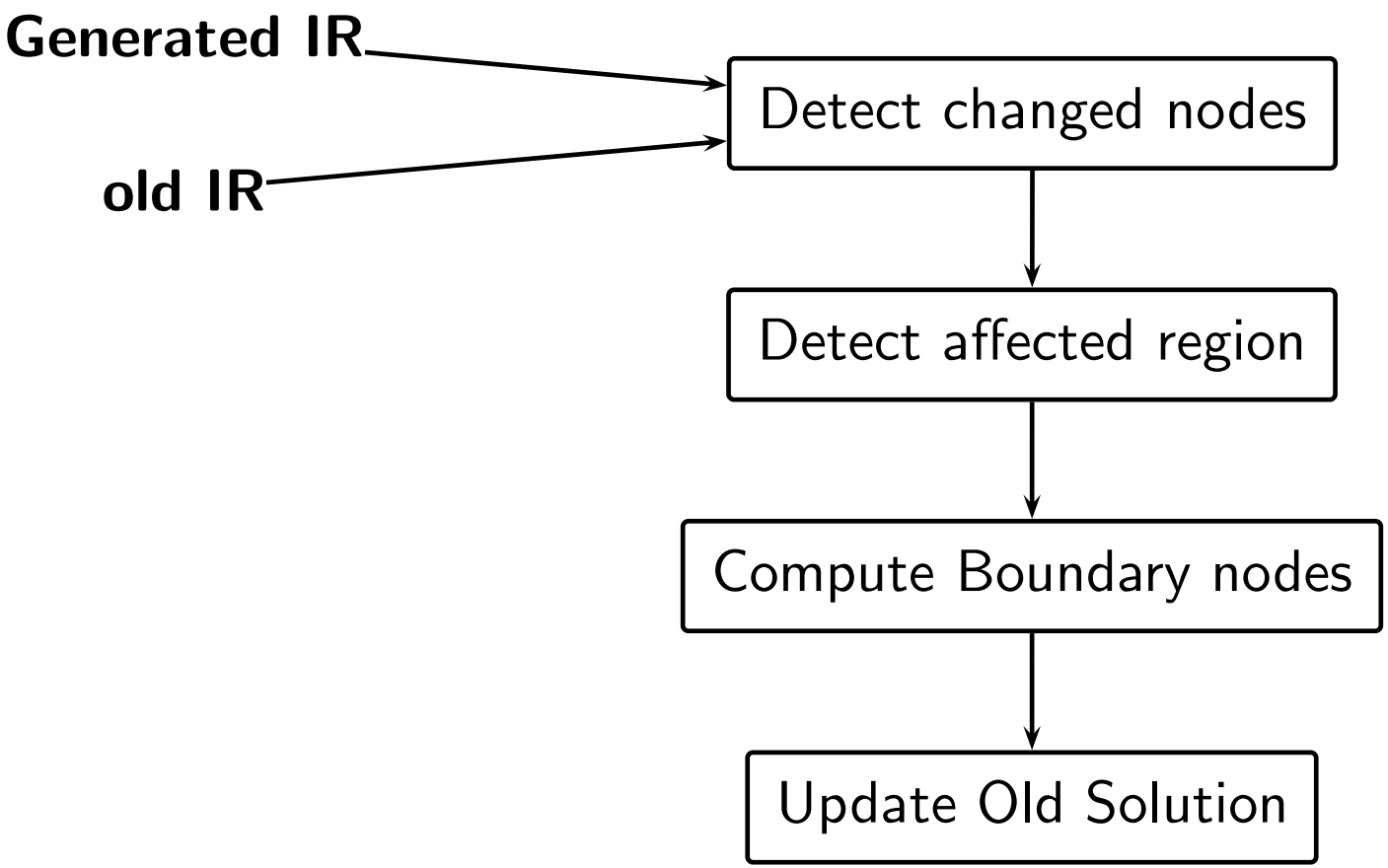
Required architecture of analyser generator



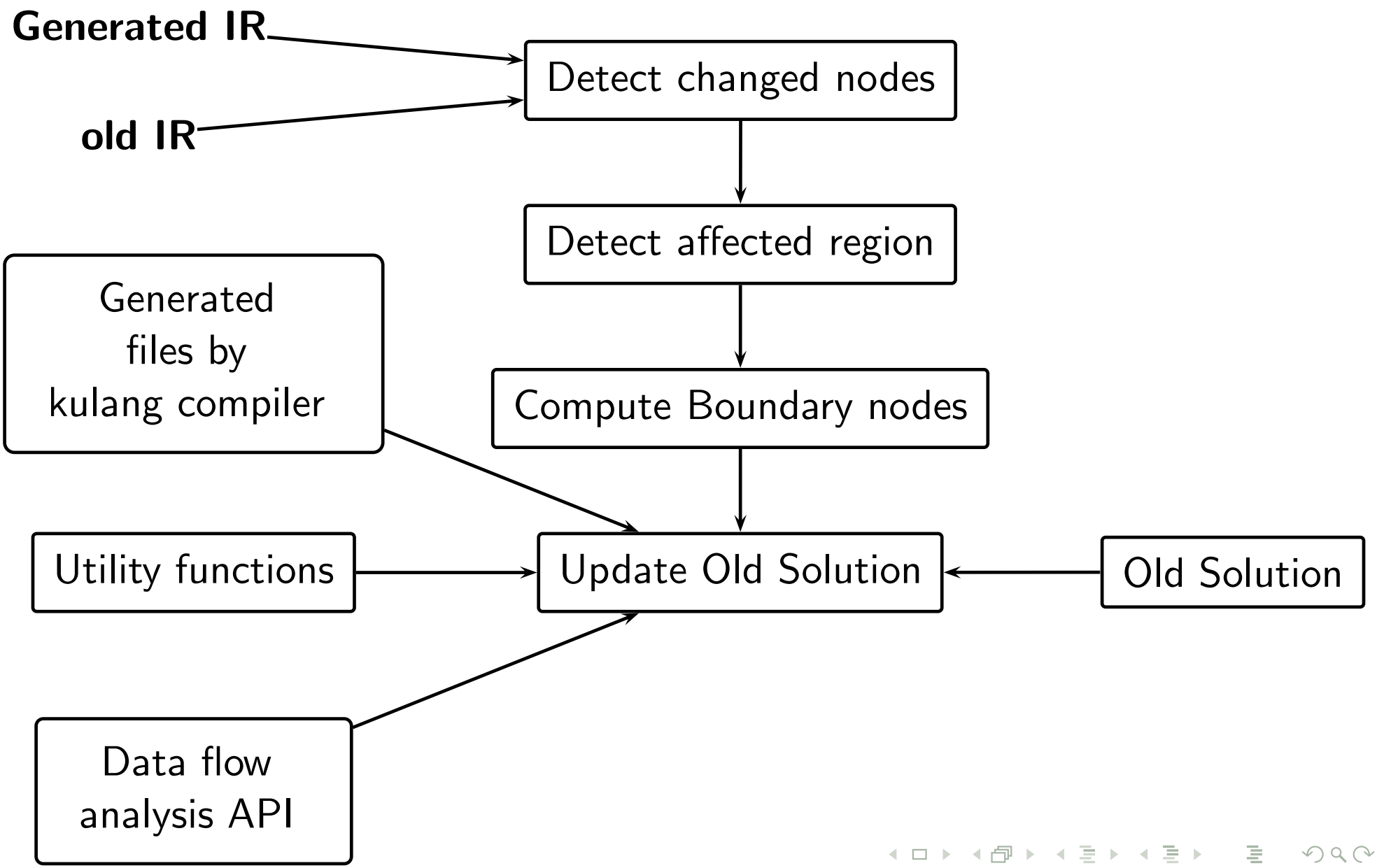
Required architecture of analyser generator



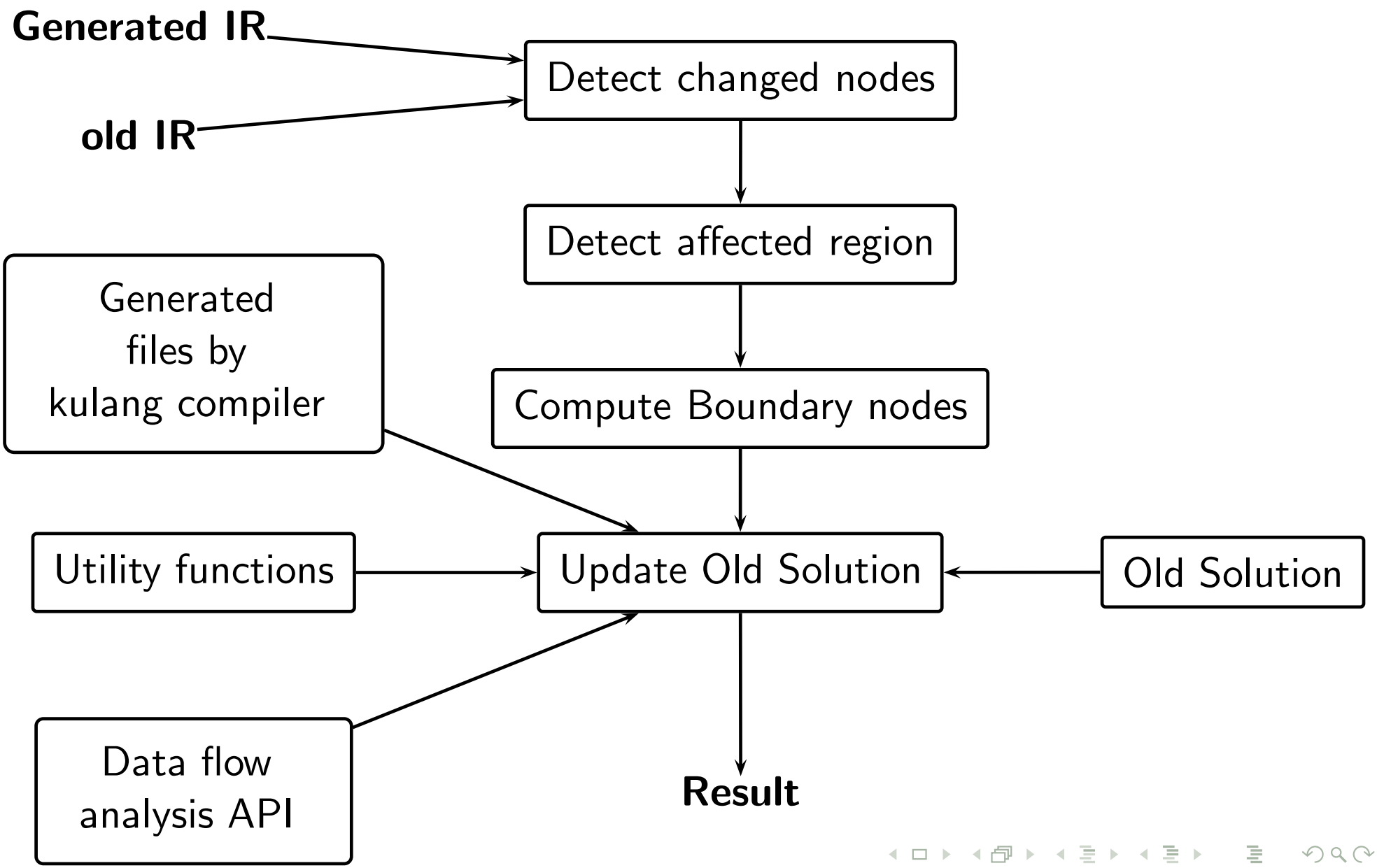
Required architecture of analyser generator



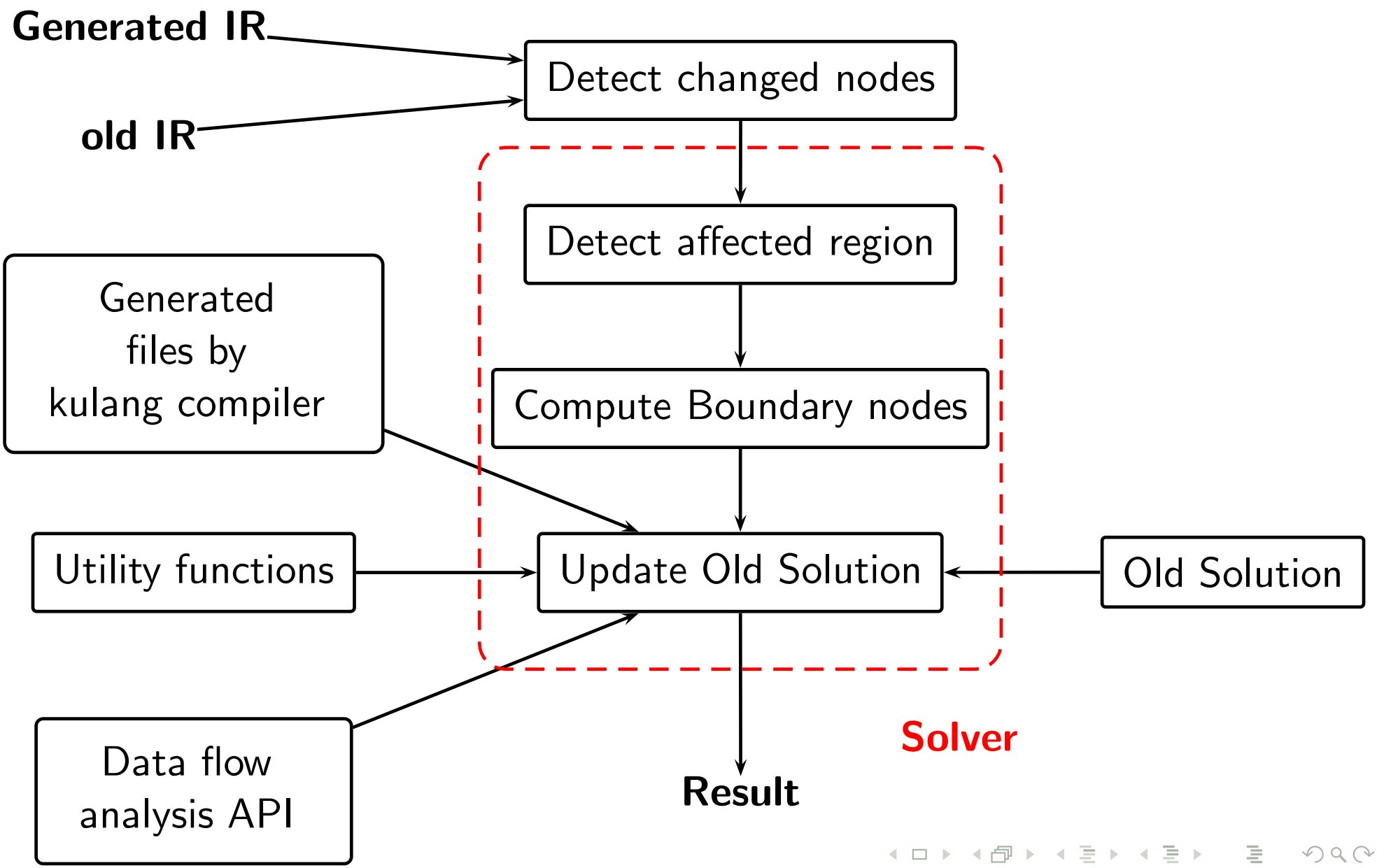
Required architecture of analyser generator



Required architecture of analyser generator



Required architecture of analyser generator



Part III

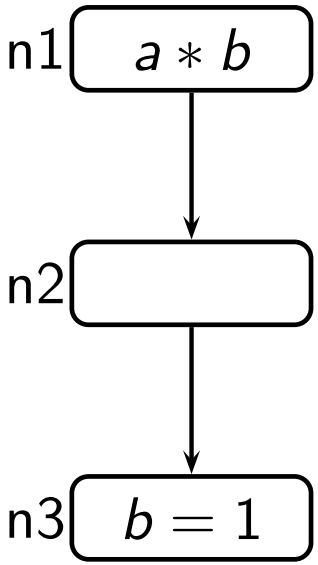
Incremental analysis in details

Flow functions in Bit-vector framework

- Possible flow functions:
  - Raise : Results is always Top
  - Lower : Results is always Bottom
  - Propagate : Propagates the value from one program point to another

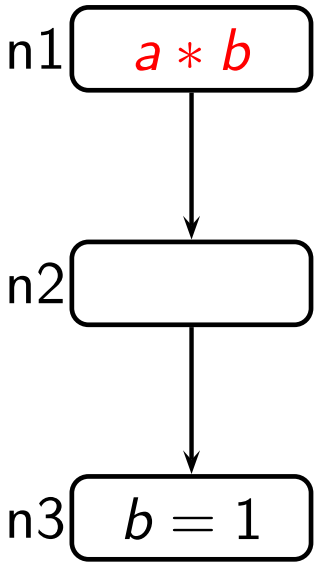
Example for flow functions

Available Expression Analysis



Example for flow functions

Available Expression Analysis



Raise Function

$\text{Gen}_1 = 1$

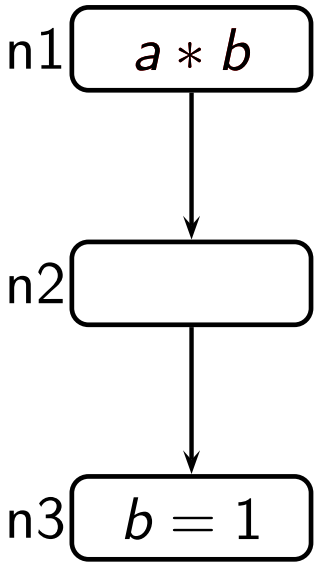
$\text{Kill}_1 = 0$

$\text{IN}_1 = 0$

$\text{OUT}_1 = \text{Gen}_1 \cup (\text{IN}_1\text{-Kill}_1) = 1$

Example for flow functions

Available Expression Analysis



Raise Function

$\text{Gen}_1 = 1$

$\text{Kill}_1 = 0$

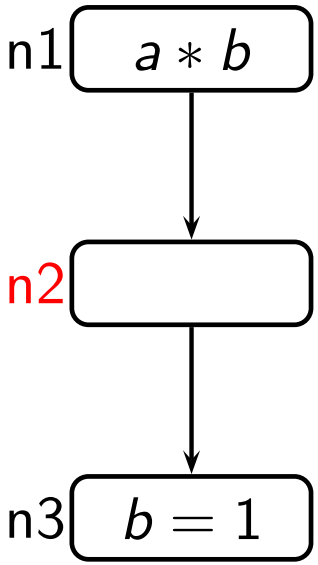
$\text{IN}_1 = 0$

$\text{OUT}_1 = \text{Gen}_1 \cup (\text{IN}_1 - \text{Kill}_1) = 1$

Result is always top

Example for flow functions

Available Expression Analysis



Propagate Function

$\text{Gen}_2 = 0$

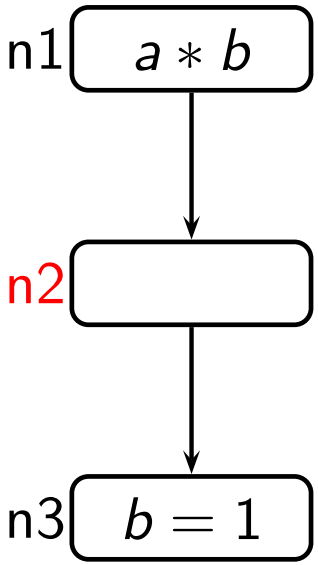
$\text{Kill}_2 = 0$

$\text{IN}_2 = 1$

$\text{OUT}_2 = \text{Gen}_2 \cup (\text{IN}_2 - \text{Kill}_2) = \text{IN}_2 = 1$

Example for flow functions

Available Expression Analysis



Propagate Function

$\text{Gen}_2 = 0$

$\text{Kill}_2 = 0$

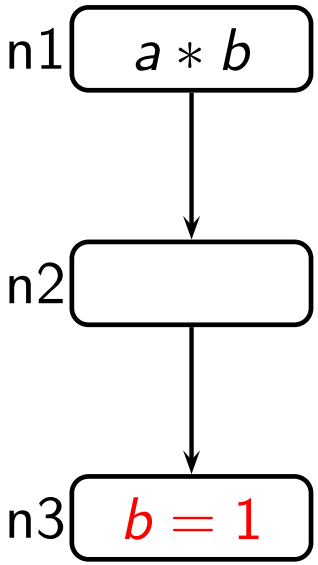
$\text{IN}_2 = 1$

$\text{OUT}_2 = \text{Gen}_2 \cup (\text{IN}_2 - \text{Kill}_2) = \text{IN}_2 = 1$

Propagates the value at IN to OUT

Example for flow functions

Available Expression Analysis



Lower Function

$\text{Gen}_3 = 0$

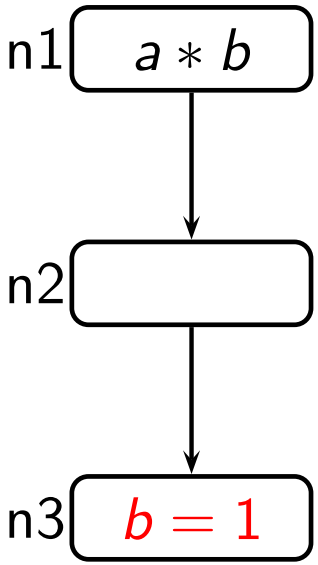
$\text{Kill}_3 = 1$

$\text{IN}_3 = 1$

$\text{OUT}_3 = \text{Gen}_3 \cup (\text{IN}_3\text{-Kill}_3) = 0$

Example for flow functions

Available Expression Analysis



Lower Function

$\text{Gen}_3 = 0$

$\text{Kill}_3 = 1$

$\text{IN}_3 = 1$

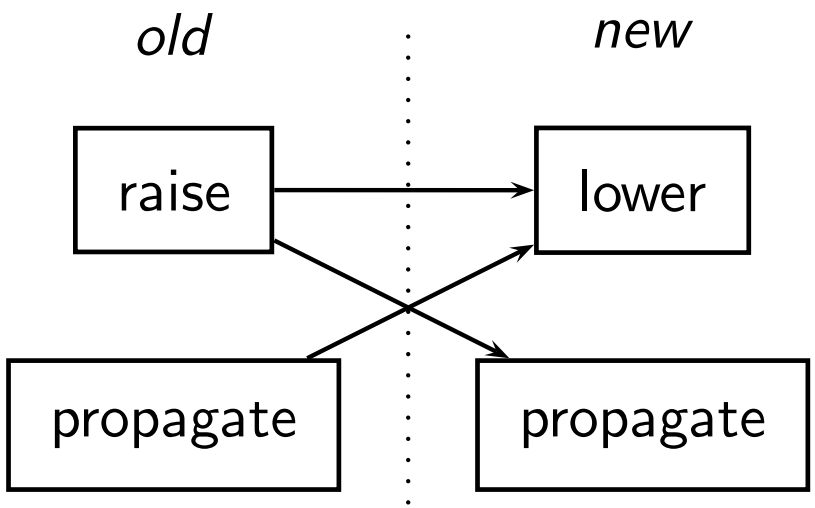
$\text{OUT}_3 = \text{Gen}_3 \cup (\text{IN}_3\text{-Kill}_3) = 0$

Result is always bottom

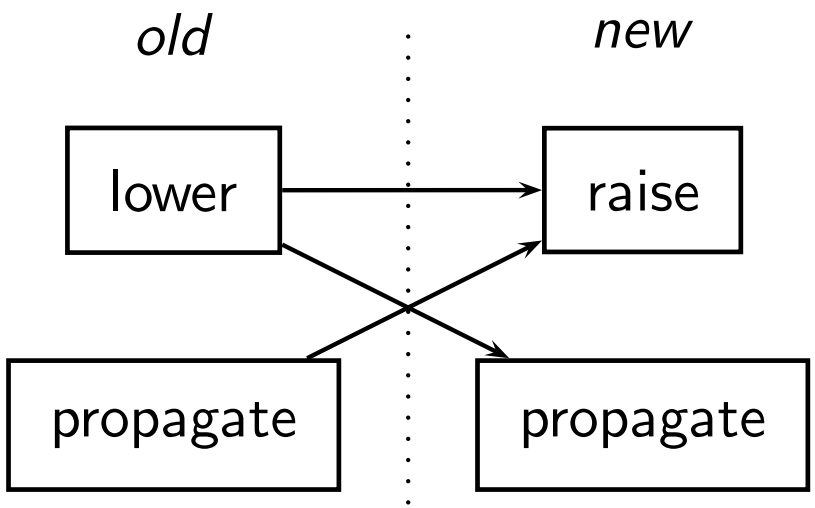
Changes in Bit-vector framework

- As a consequence of some change in a node, some data flow values may:
  - change from top to bottom
  - change from bottom to top
  - remain same

Possible changes in flow functions for top to bottom change



Possible changes in flow functions for bottom to top change

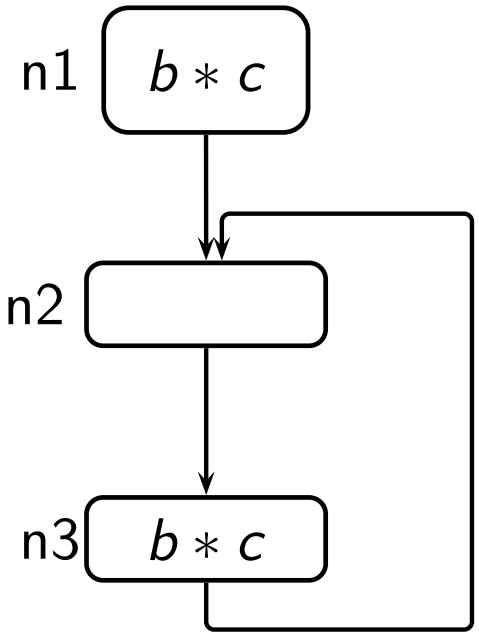


Handling Top to Bottom change

- Top value is an intermediate value until data flow analysis is completed
- Whenever there is top to bottom change, the changes can be propagated directly to its neighbouring nodes

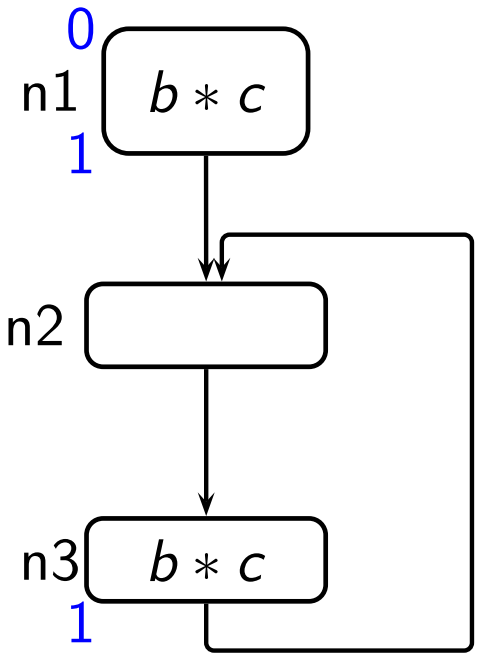
Example for Top to Bottom change

Initial Available Expression Analysis



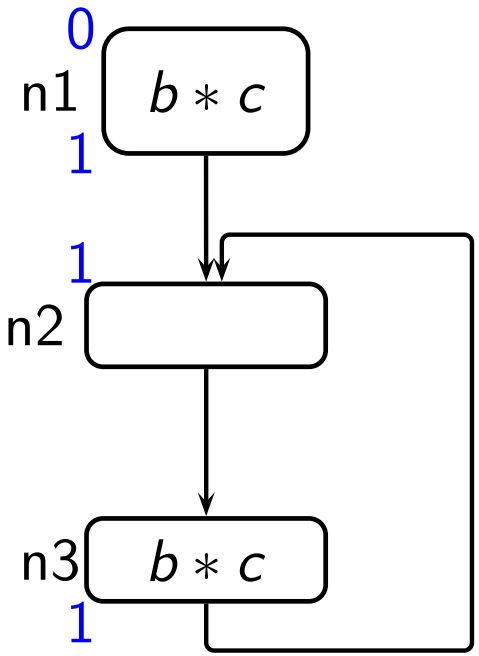
Example for Top to Bottom change

Initial Available Expression Analysis



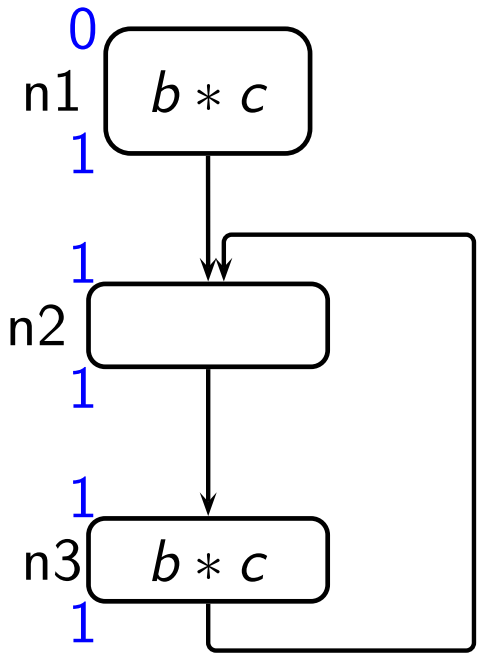
Example for Top to Bottom change

Initial Available Expression Analysis

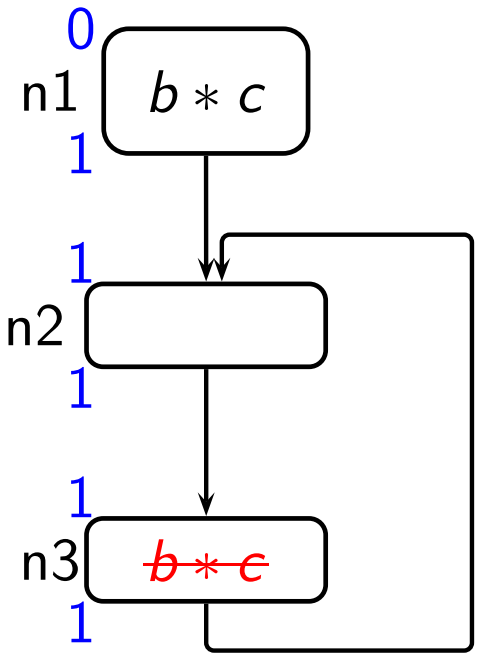


Example for Top to Bottom change

Initial Available Expression Analysis

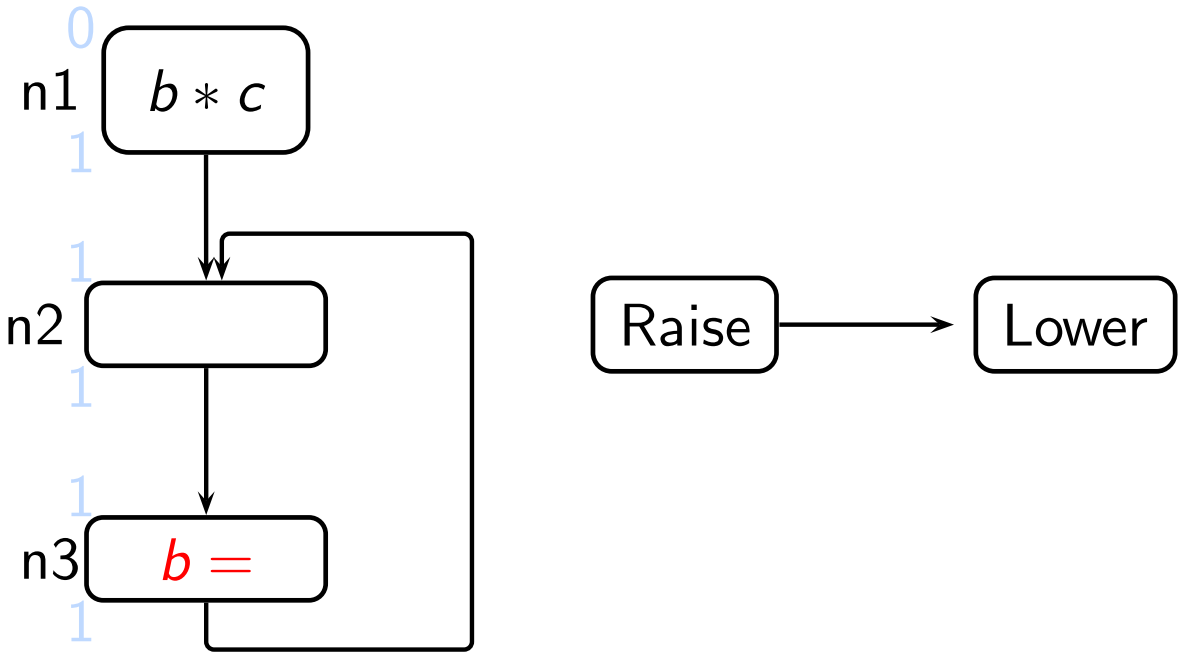


Example for Top to Bottom change



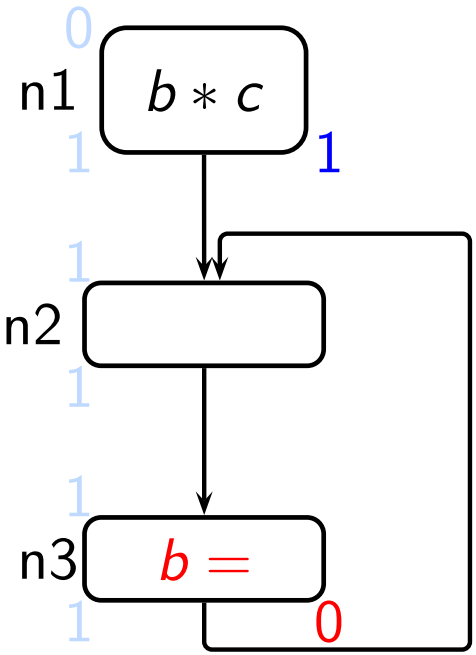
Example for Top to Bottom change

Top to Bottom change

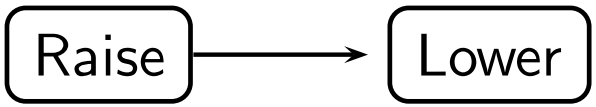


Example for Top to Bottom change

Top to Bottom change



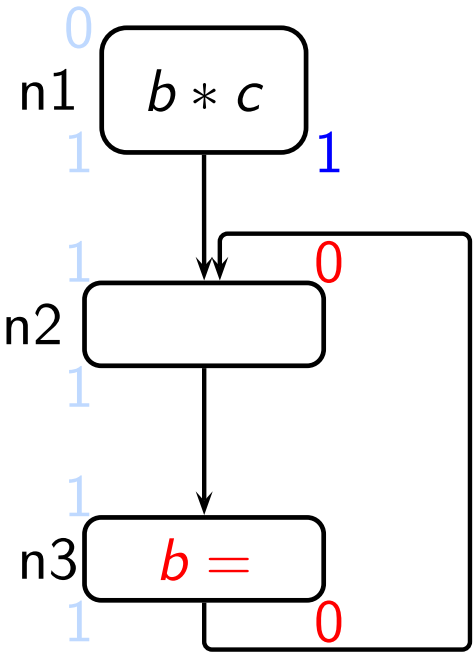
$IN_2 = OUT_1 \cap OUT_3$



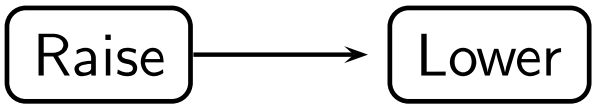
Directly Propagate the change to its neighbour

Example for Top to Bottom change

Top to Bottom change



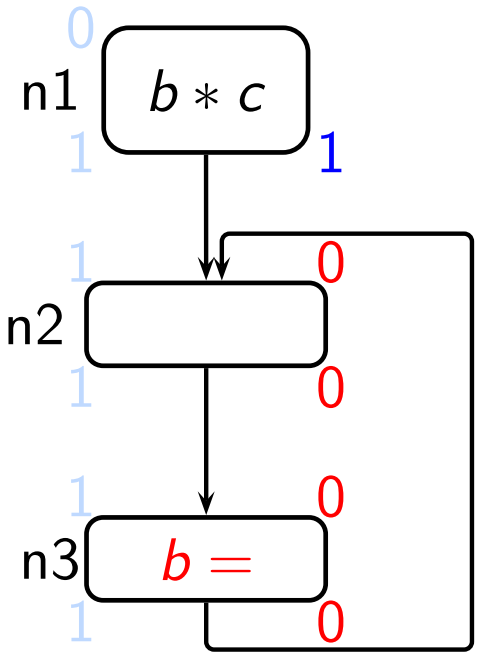
$IN_2 = OUT_1 \cap OUT_3$



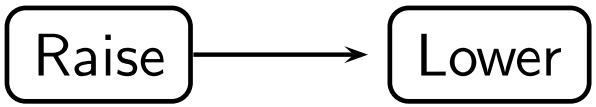
Directly Propagate the change to its neighbour

Example for Top to Bottom change

Top to Bottom change



$IN_2 = OUT_1 \cap OUT_3$



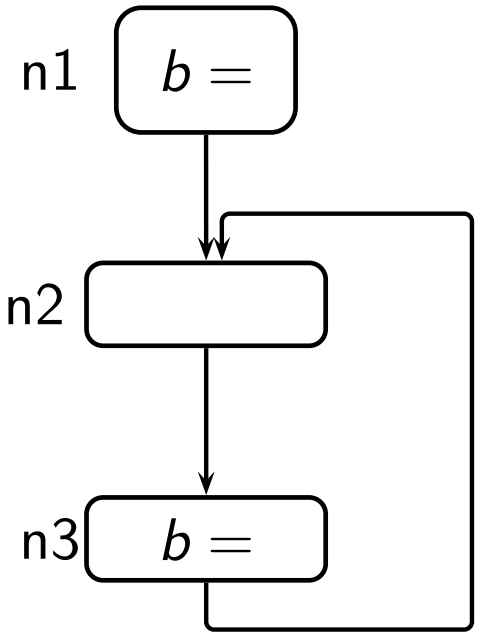
Directly Propagate the change to its neighbour

Handling Bottom to Top change

- Bottom value is a final value even during analysis
- Whenever there is bottom to top change, we cannot directly propagate the changes to its neighbouring nodes
- Need some more processing

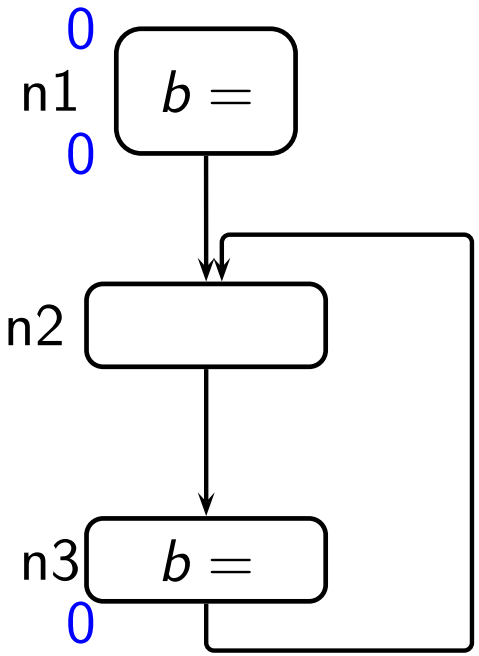
Example for Bottom to Top change

Initial Available Expression Analysis



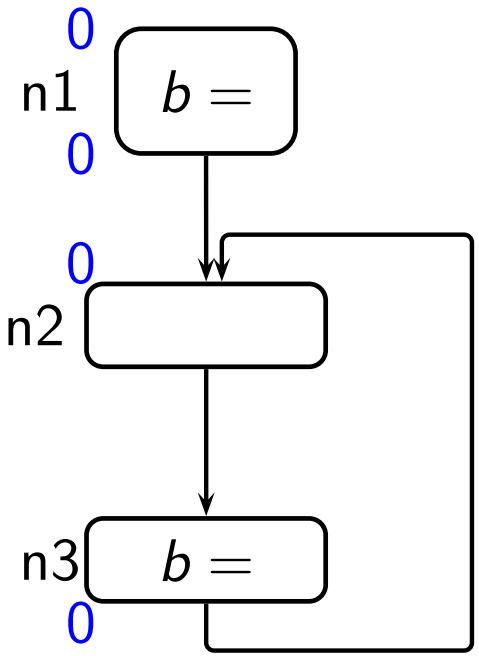
Example for Bottom to Top change

Initial Available Expression Analysis



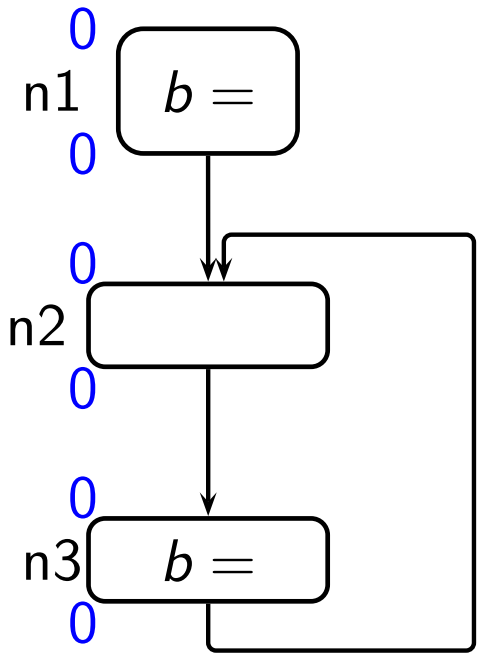
Example for Bottom to Top change

Initial Available Expression Analysis

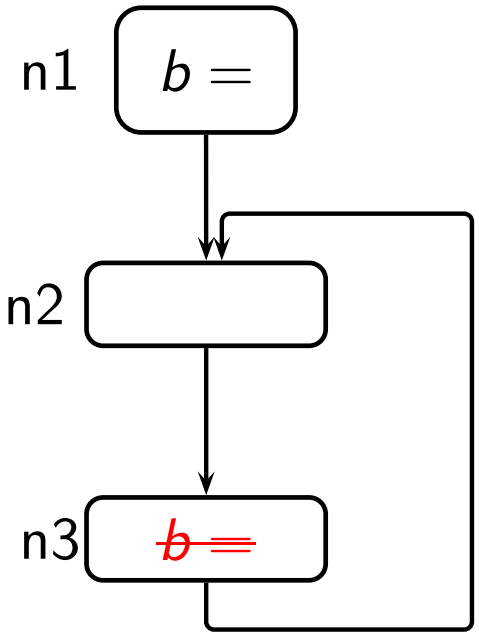


Example for Bottom to Top change

Initial Available Expression Analysis

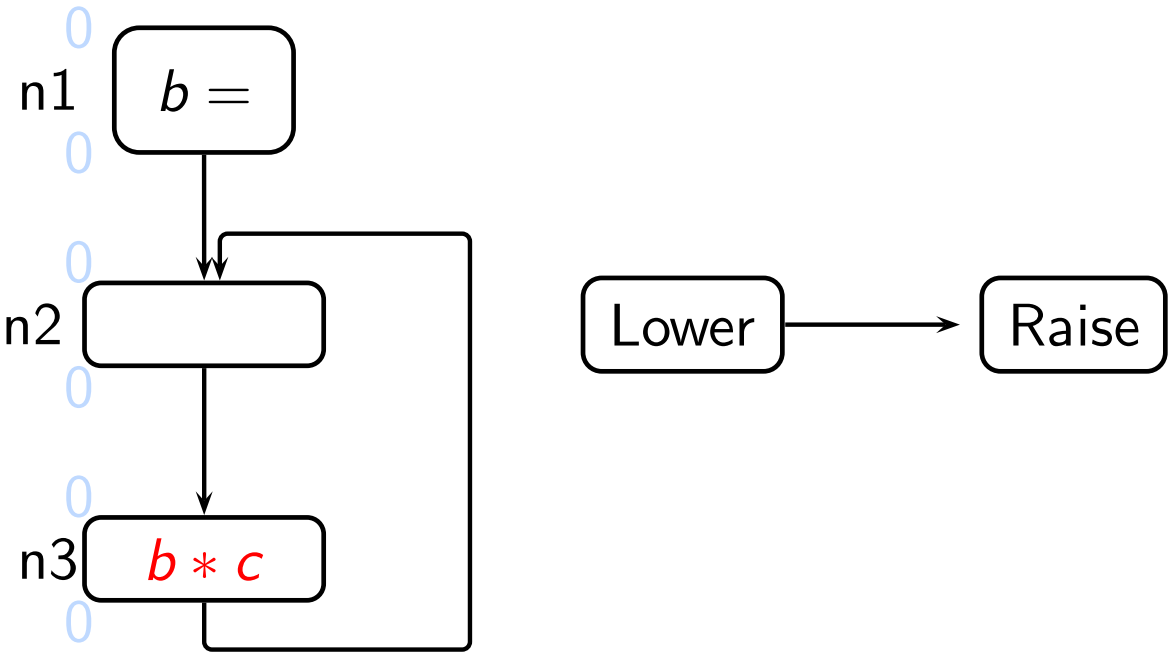


Example for Bottom to Top change



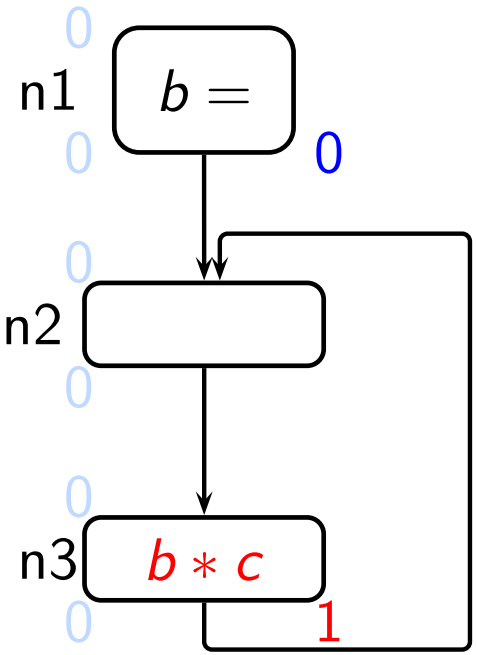
Example for Bottom to Top change

Bottom to Top change

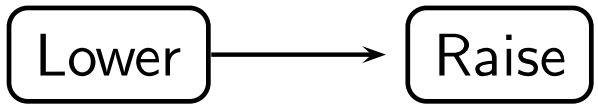


Example for Bottom to Top change

Bottom to Top change



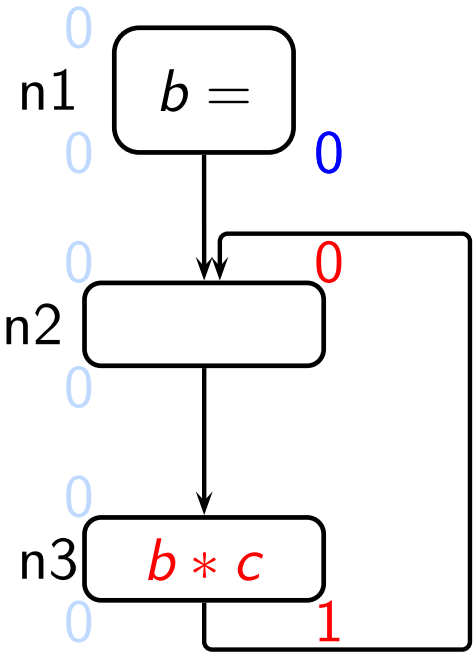
$IN_2 = OUT_1 \cap OUT_3$



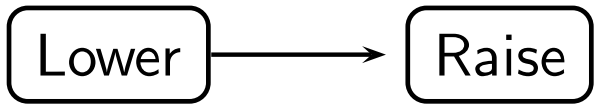
Cannot propagate the change to its neighbouring nodes

Example for Bottom to Top change

Bottom to Top change



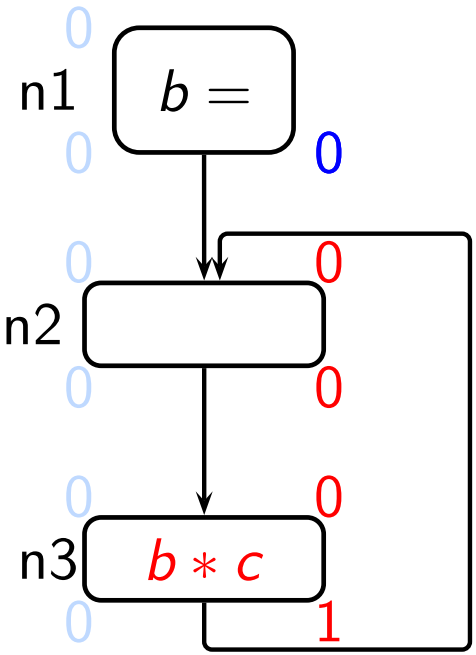
$IN_2 = OUT_1 \cap OUT_3$



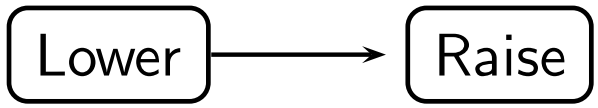
Cannot propagate the change to its neighbouring nodes

Example for Bottom to Top change

Bottom to Top change



$IN_2 = OUT_1 \cap OUT_3$



Cannot propagate the change to its neighbouring nodes

Bottom to Top change

- Need some more processing

Bottom to Top change

- Steps to incorporate Bottom to Top change:

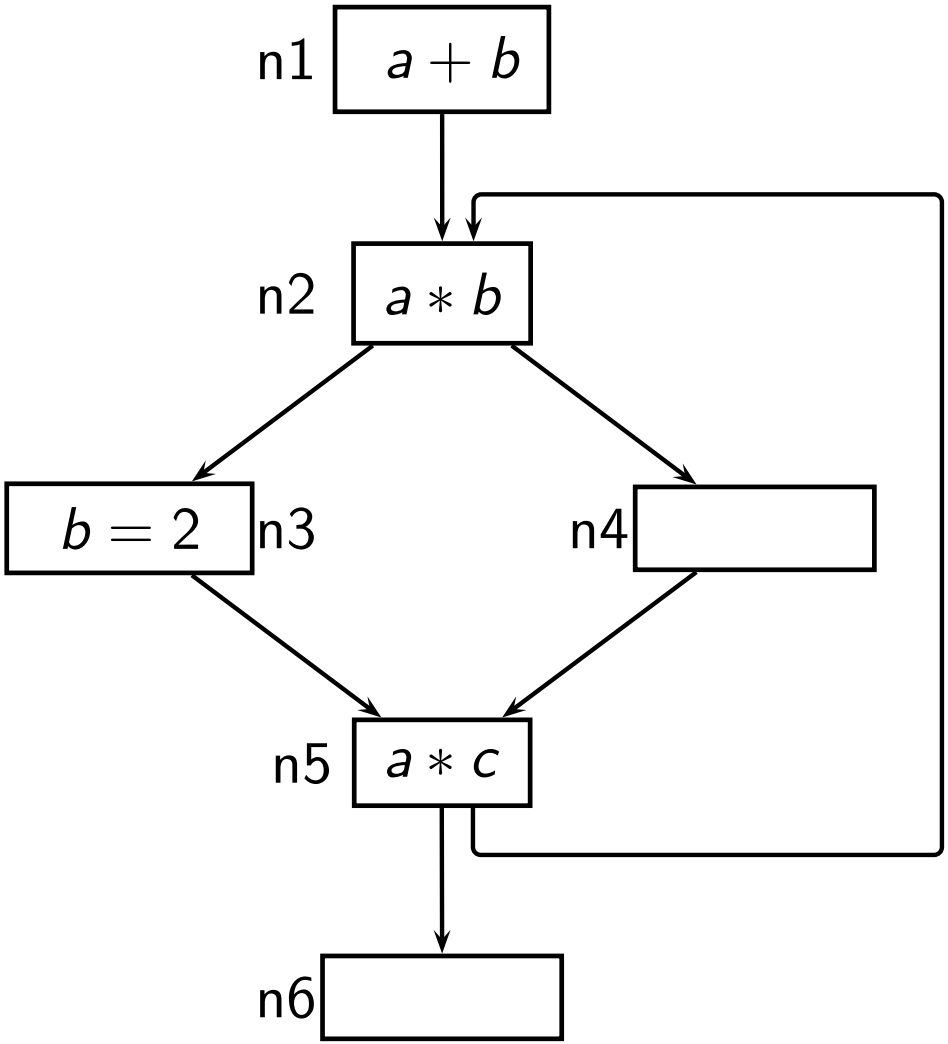
Bottom to Top change

- Steps to incorporate Bottom to Top change:
  - Identify the data flow values which may becomes top

Bottom to Top change

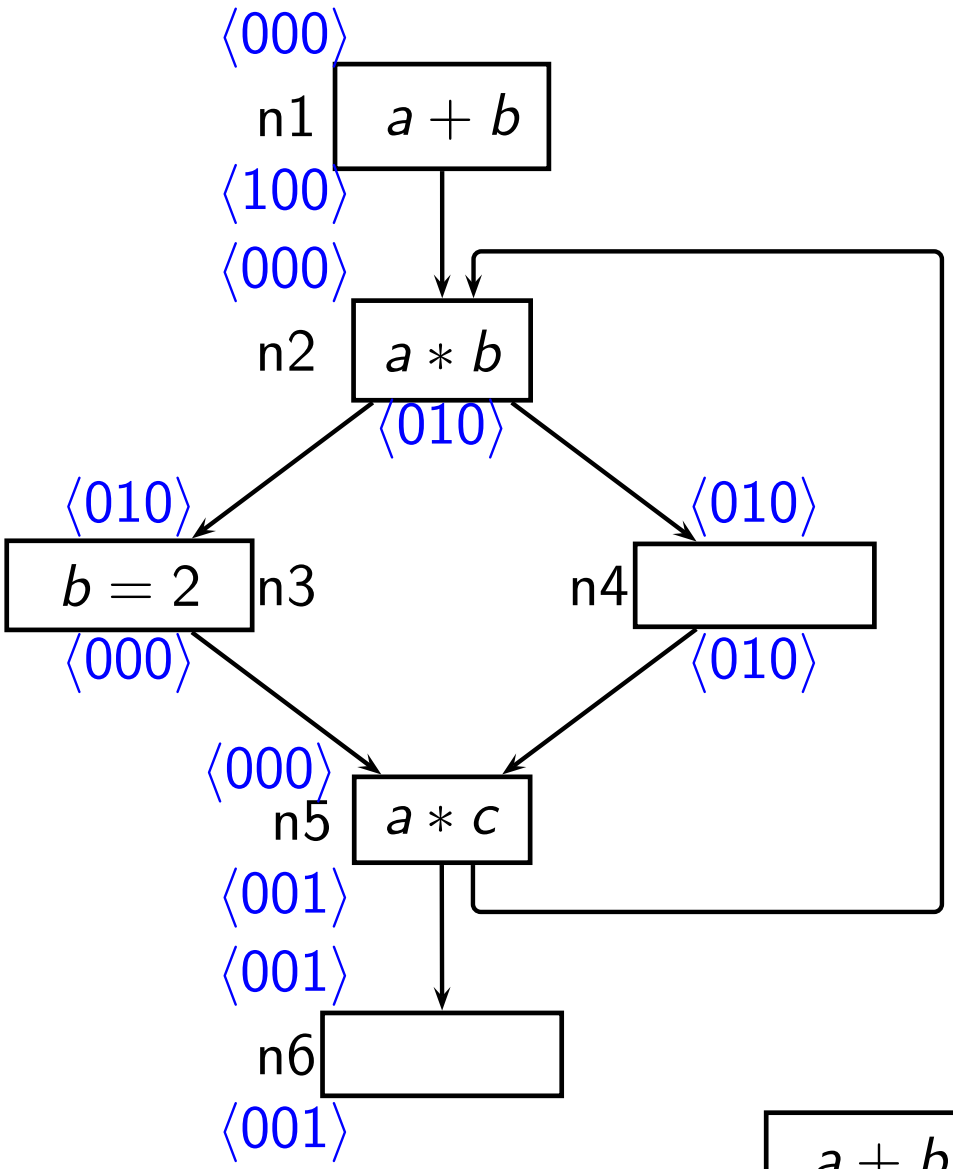
- Steps to incorporate Bottom to Top change:
  - Identify the data flow values which may becomes top
  - Find out the data flow values which must remain bottom due to the effect of some other property

Motivating Example



$a + b$	$a * b$	$a * c$

Motivating Example

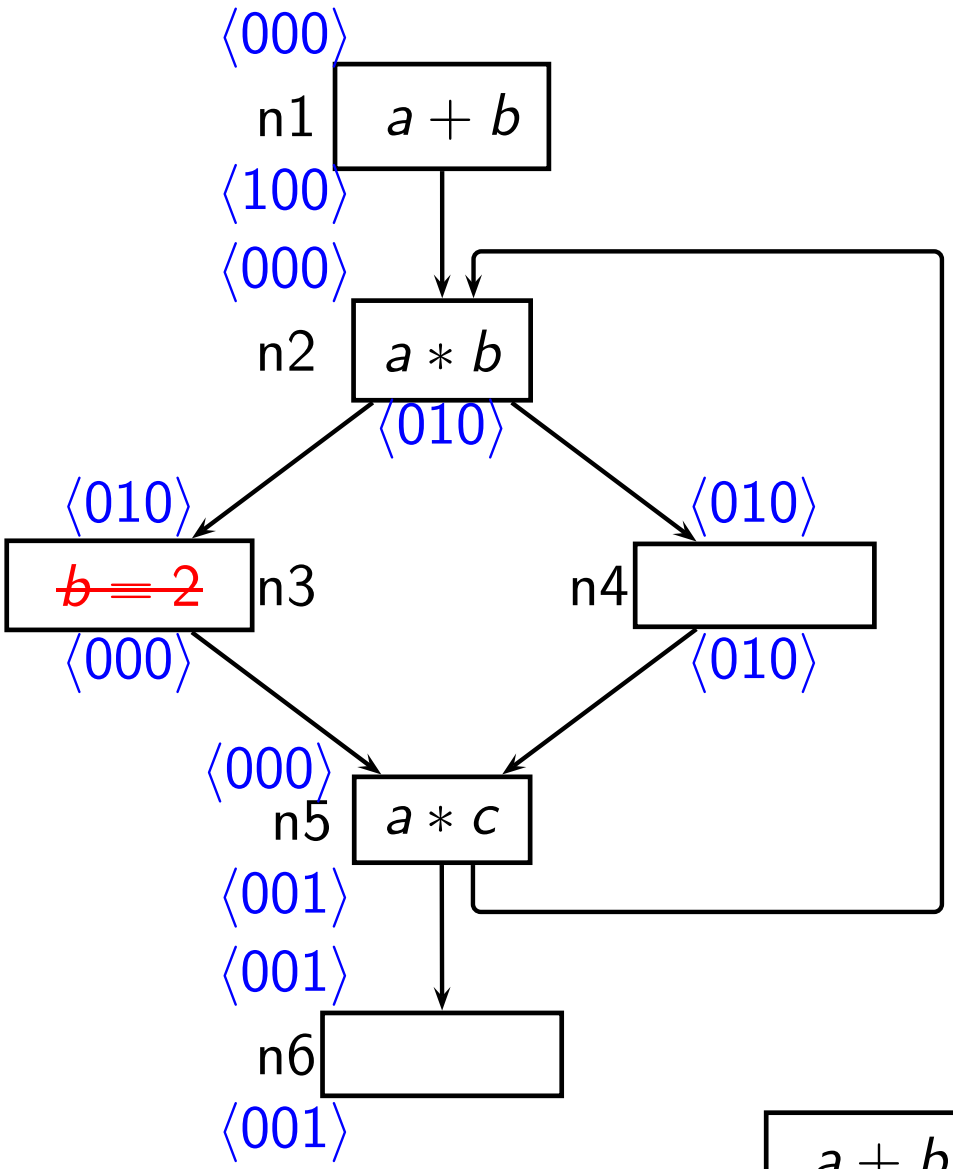


Initial Available Expression Analysis

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	0	0	0	1	0	0
3.	0	0	1	0	0	0
4.	0	0	1	1	0	0
5.	0	0	0	0	0	1
6.	0	0	0	0	1	1

$a+b$	$a*b$	$a*c$

Motivating Example

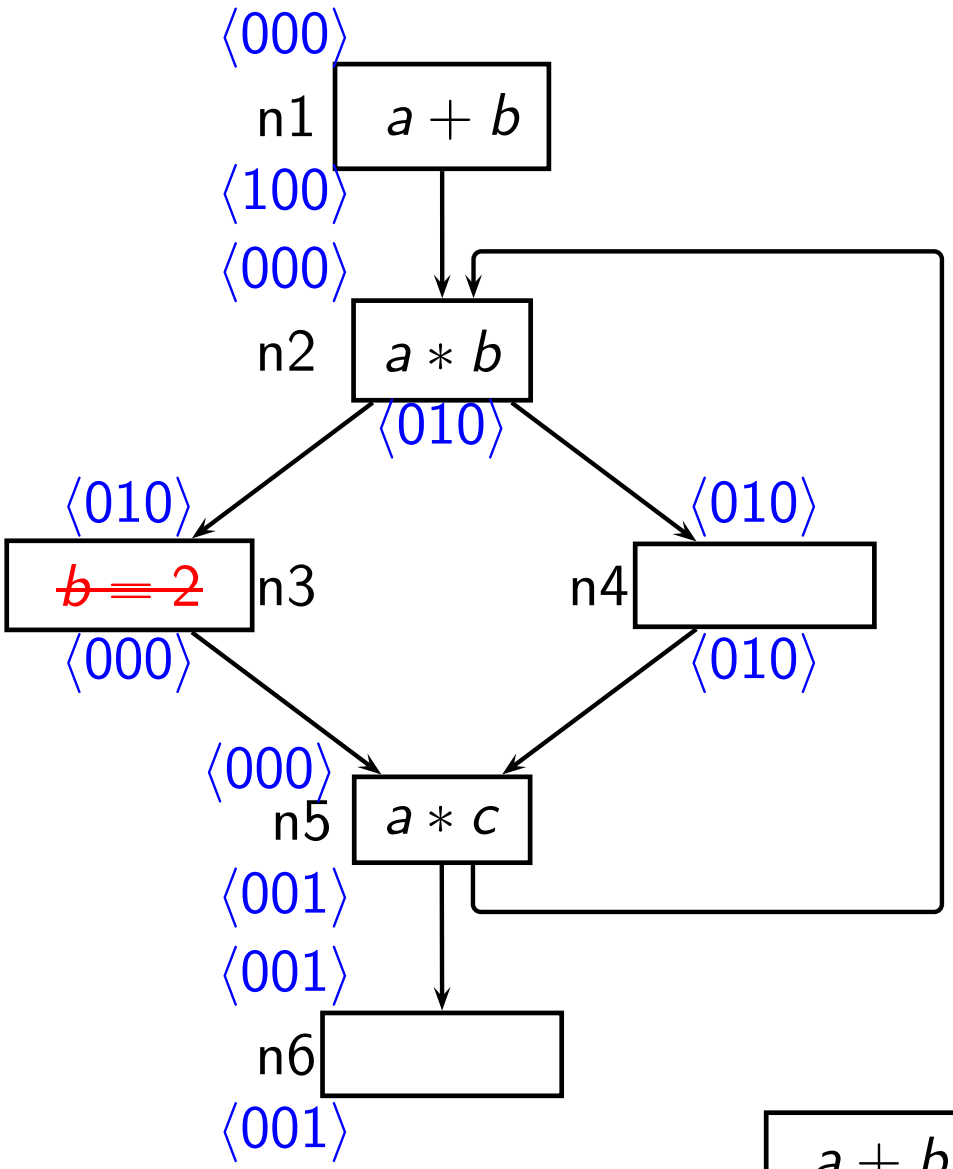


Initial Available Expression Analysis

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	0	0	0	1	0	0
3.	0	0	1	0	0	0
4.	0	0	1	1	0	0
5.	0	0	0	0	0	1
6.	0	0	0	0	1	1

$a+b$	$a*b$	$a*c$

Motivating Example



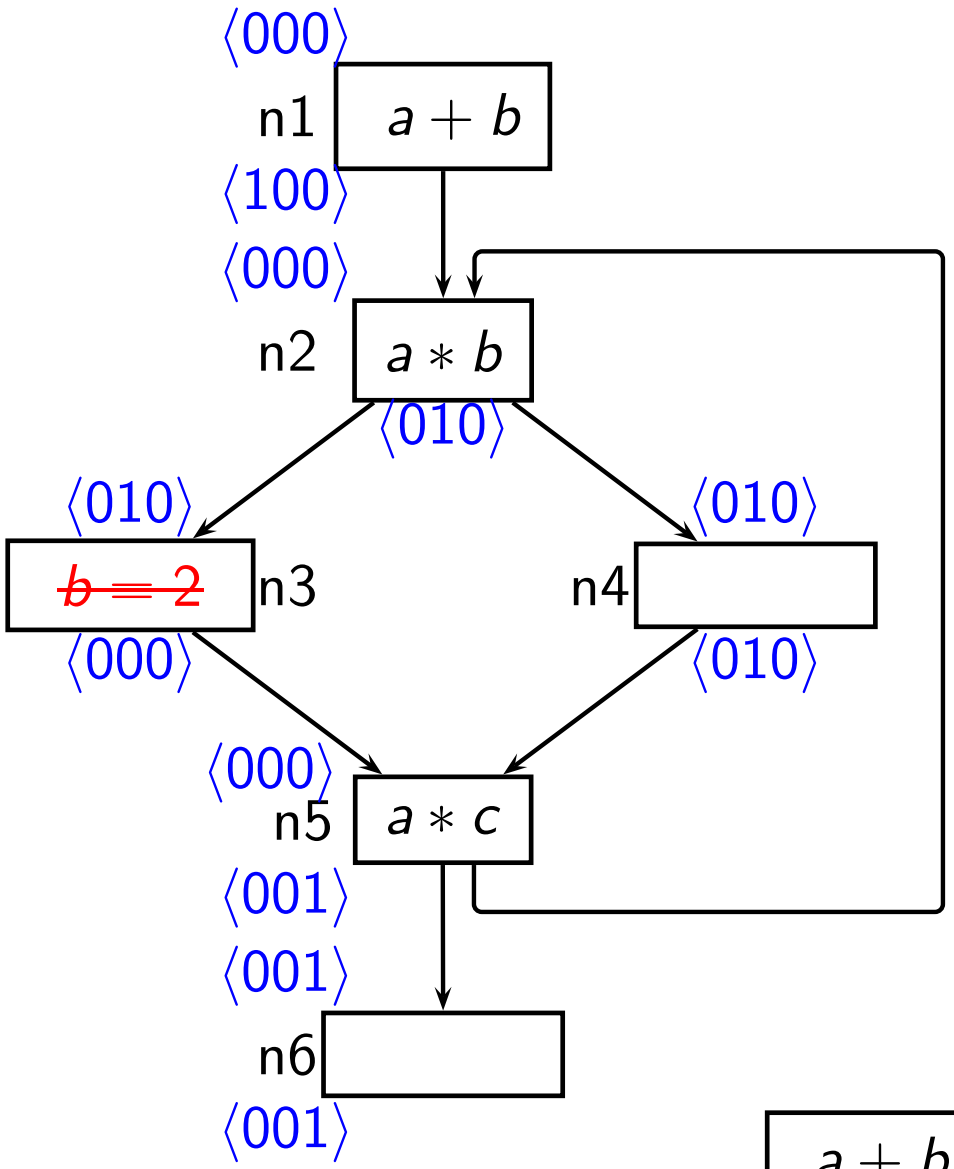
Initial Available Expression Analysis

lower

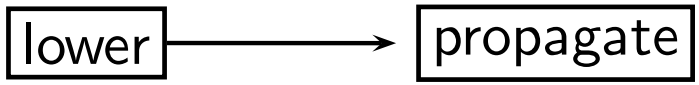
propagate

$a+b$	$a*b$	$a*c$

Motivating Example

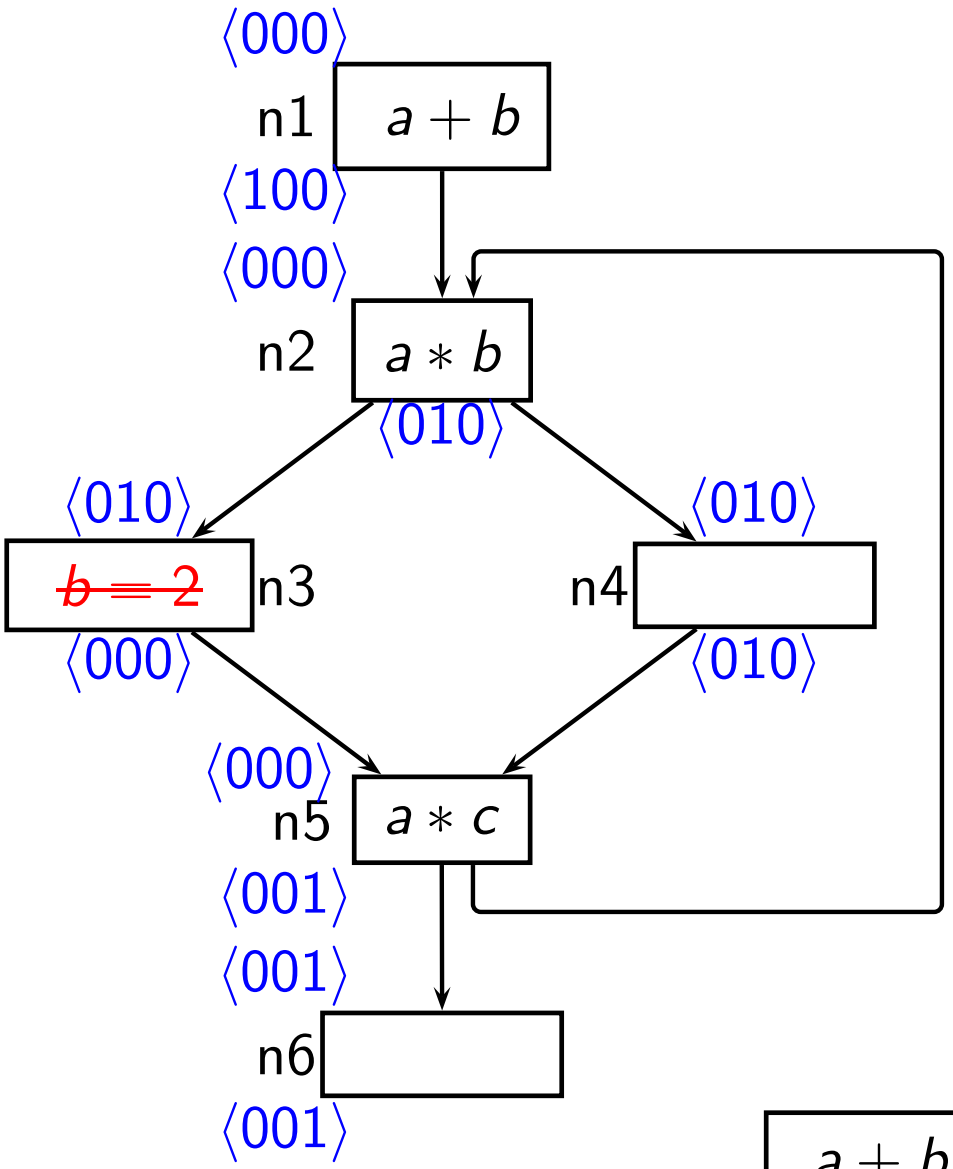


Initial Available Expression Analysis

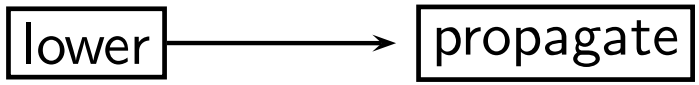


$a + b$	$a * b$	$a * c$

Motivating Example



Initial Available Expression Analysis



Bottom to Top change

$a + b$	$a * b$	$a * c$

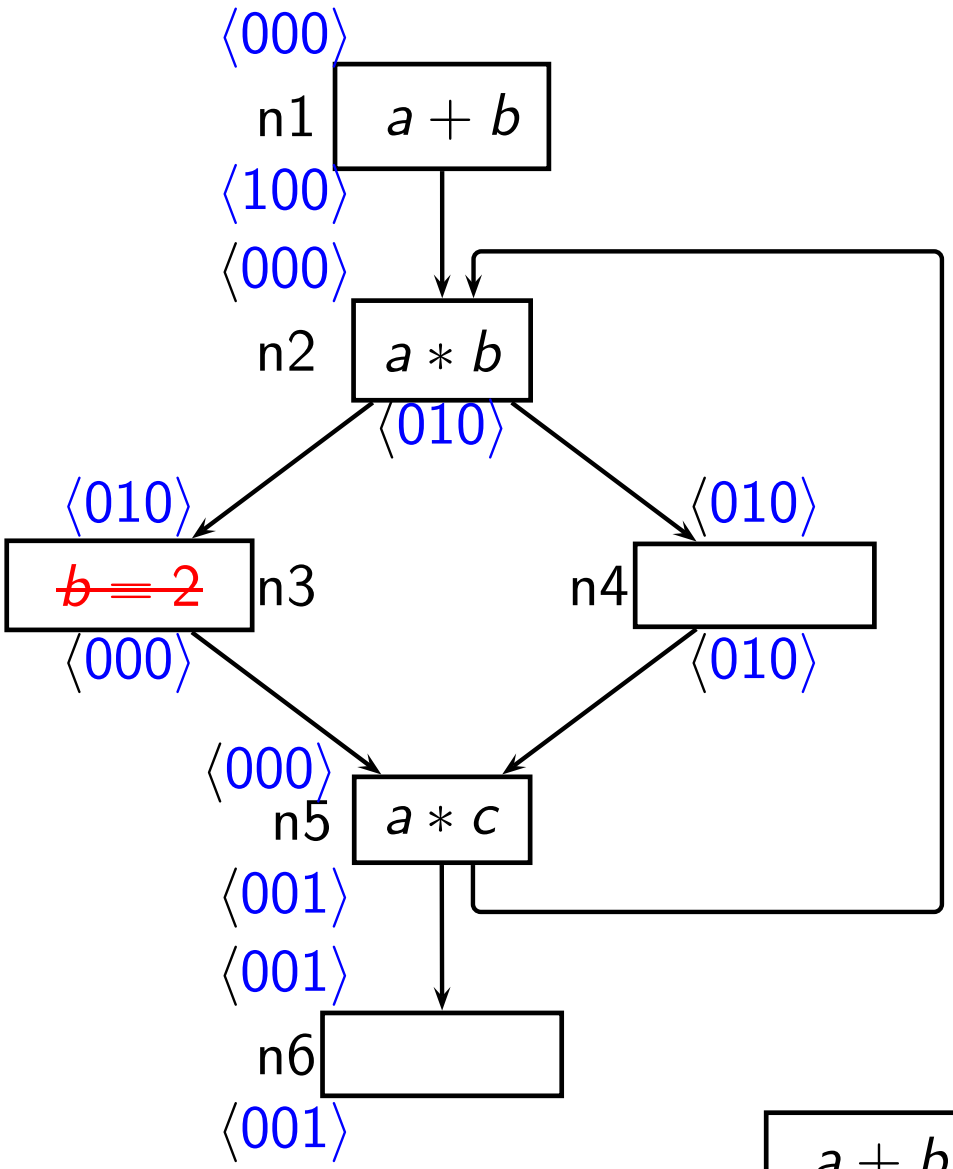
Motivating Example - Step 1

- The data flow values which were 0 and *may* become 1 due to this change

Motivating Example - Step 1

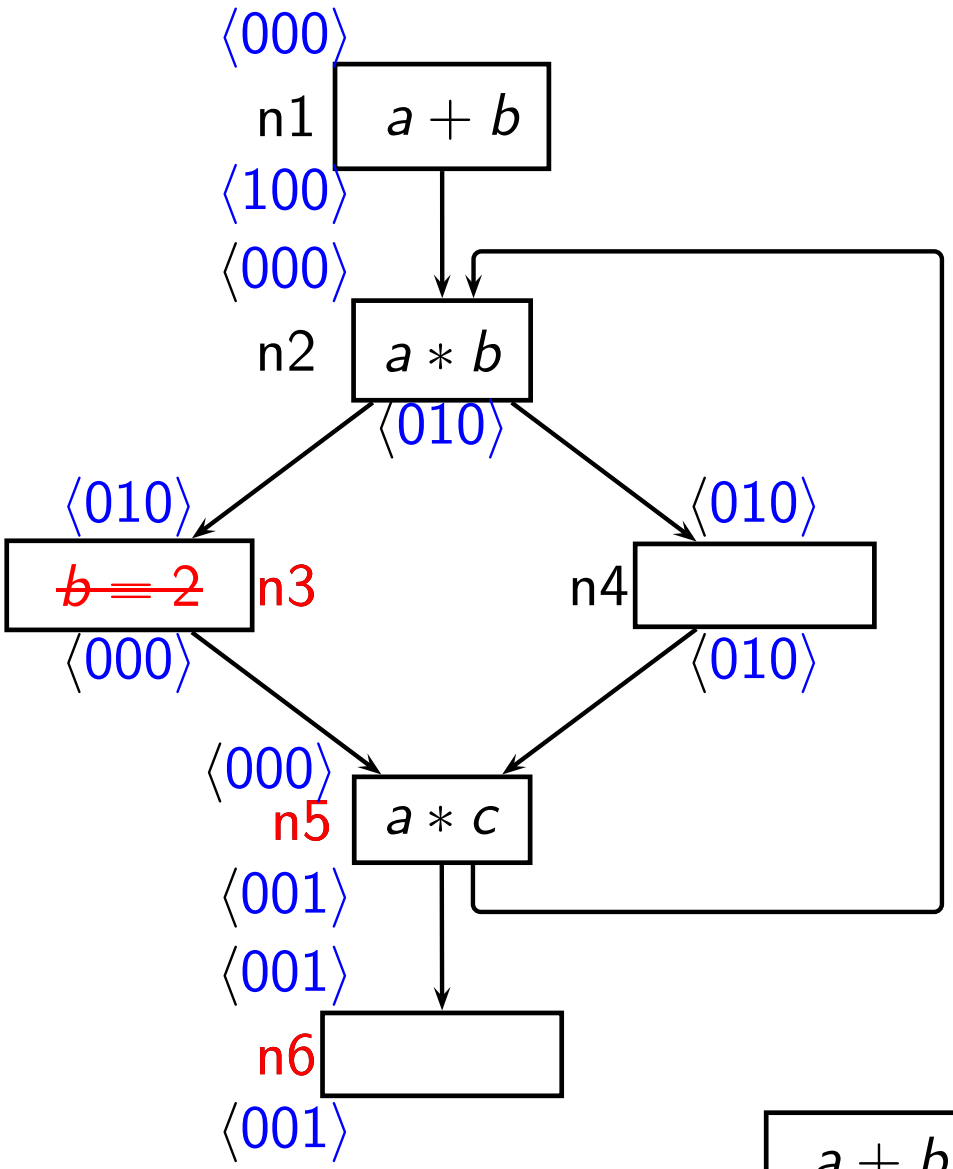
- The data flow values which were 0 and *may* become 1 due to this change
  - Affected region

Motivating Example - Step 1



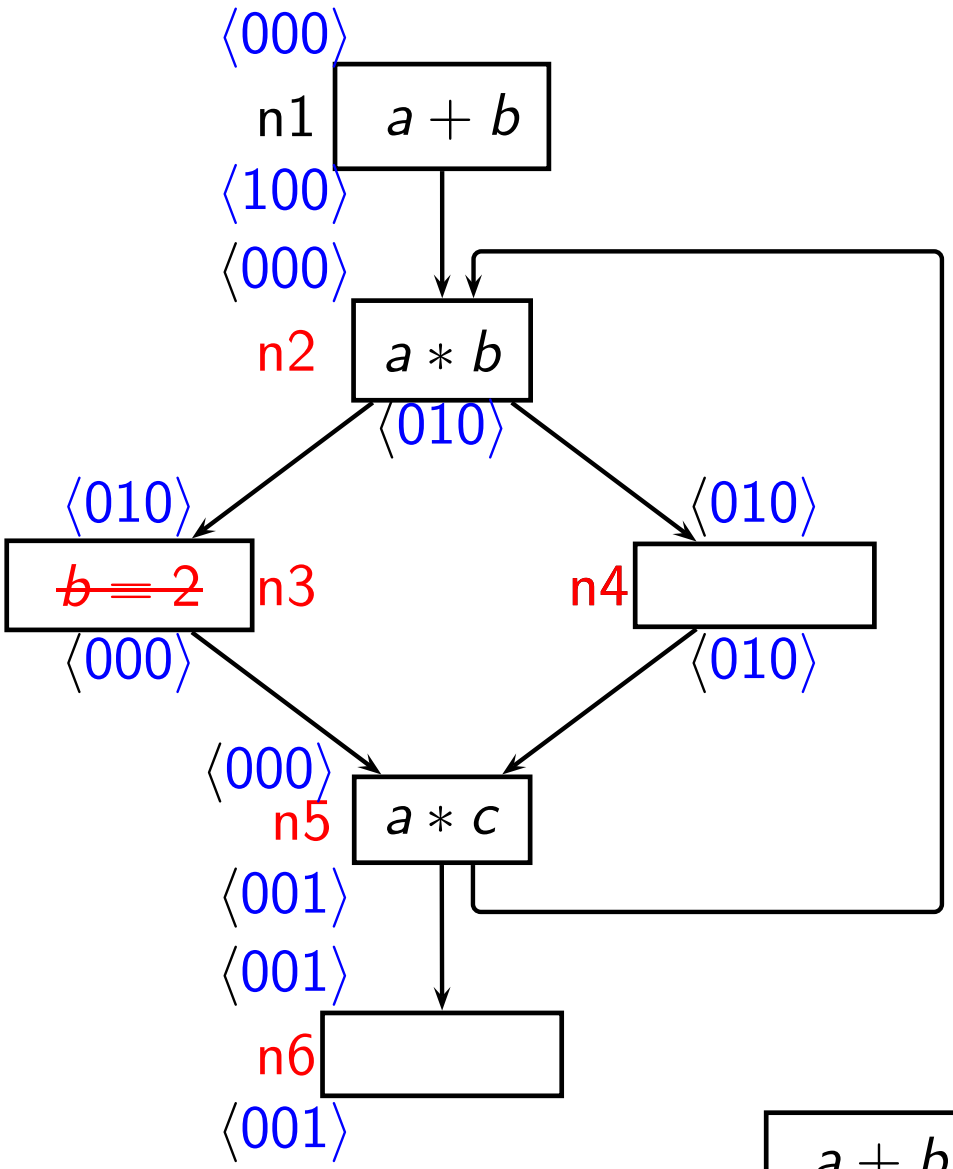
$a + b$	$a * b$	$a * c$

Motivating Example - Step 1



$a + b$	$a * b$	$a * c$

Motivating Example - Step 1

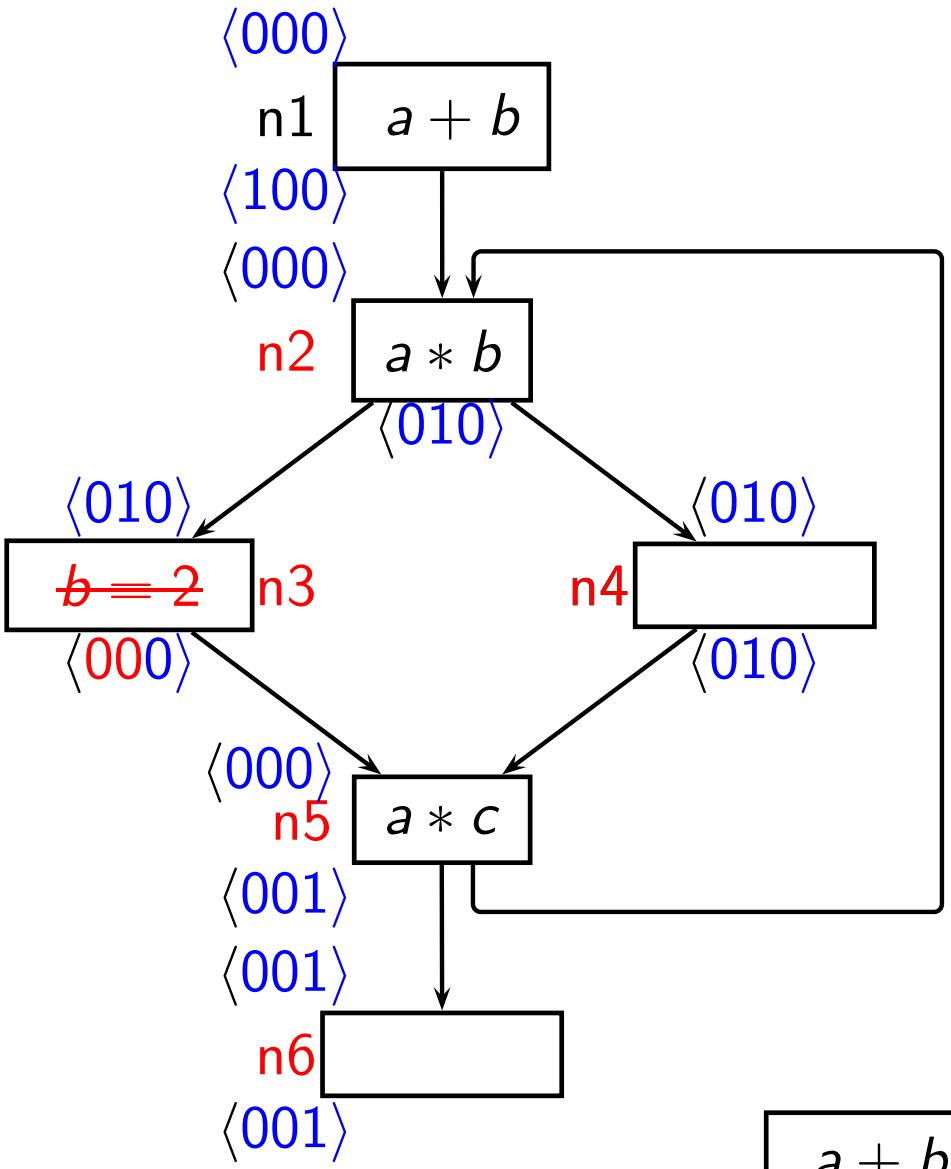


Affected Region

$\langle$ OUT_3 IN_5 OUT_5, IN_6, OUT_6, IN_2, OUT_2, IN_4, OUT_4, $IN_3\rangle$

$a + b$	$a * b$	$a * c$

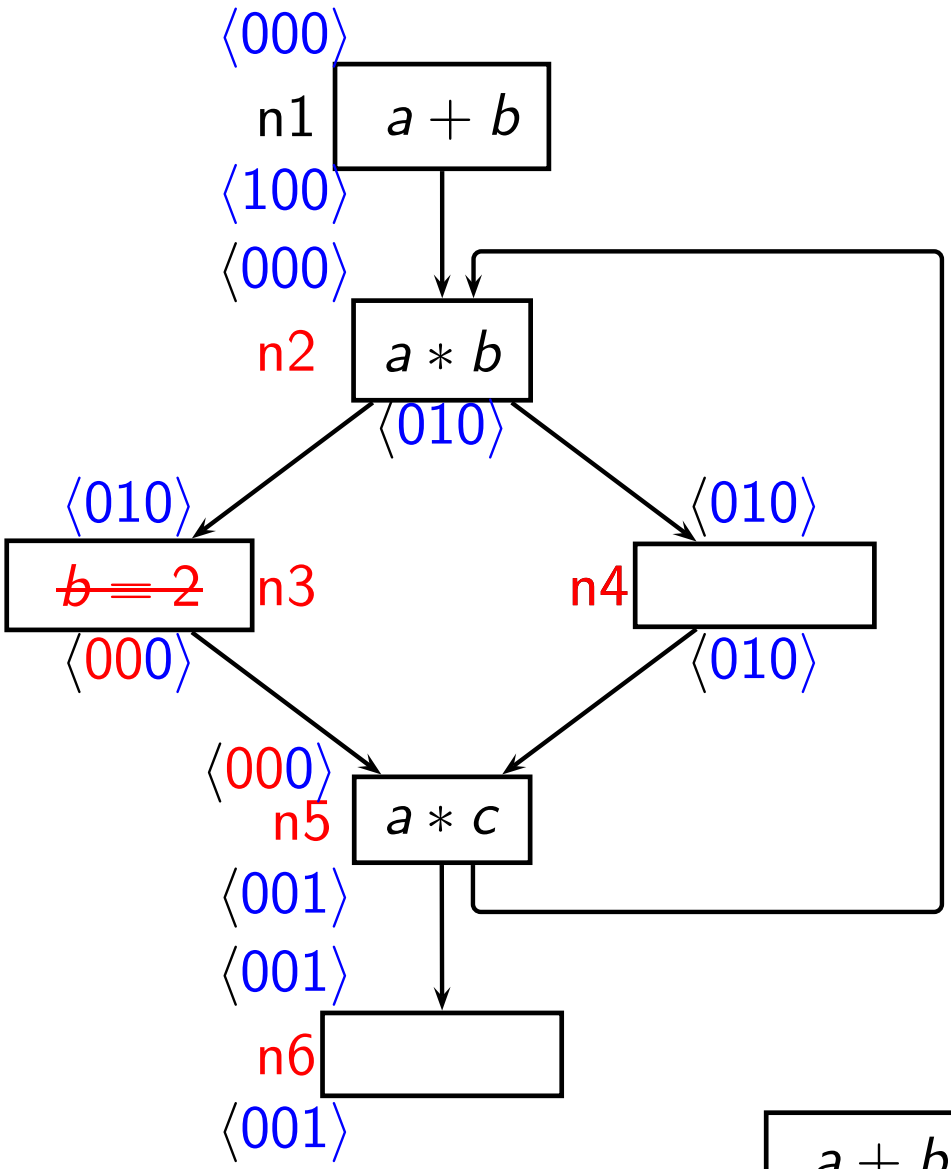
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

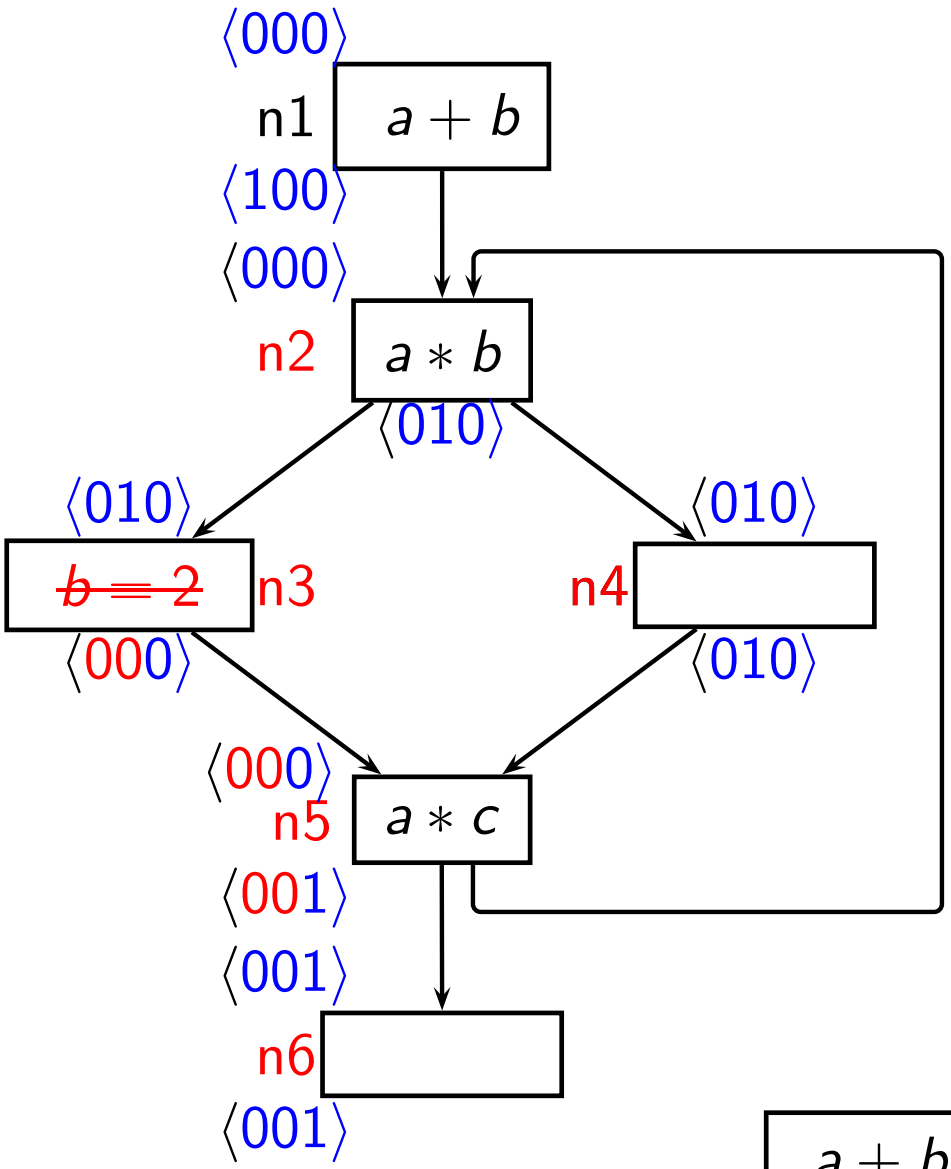
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

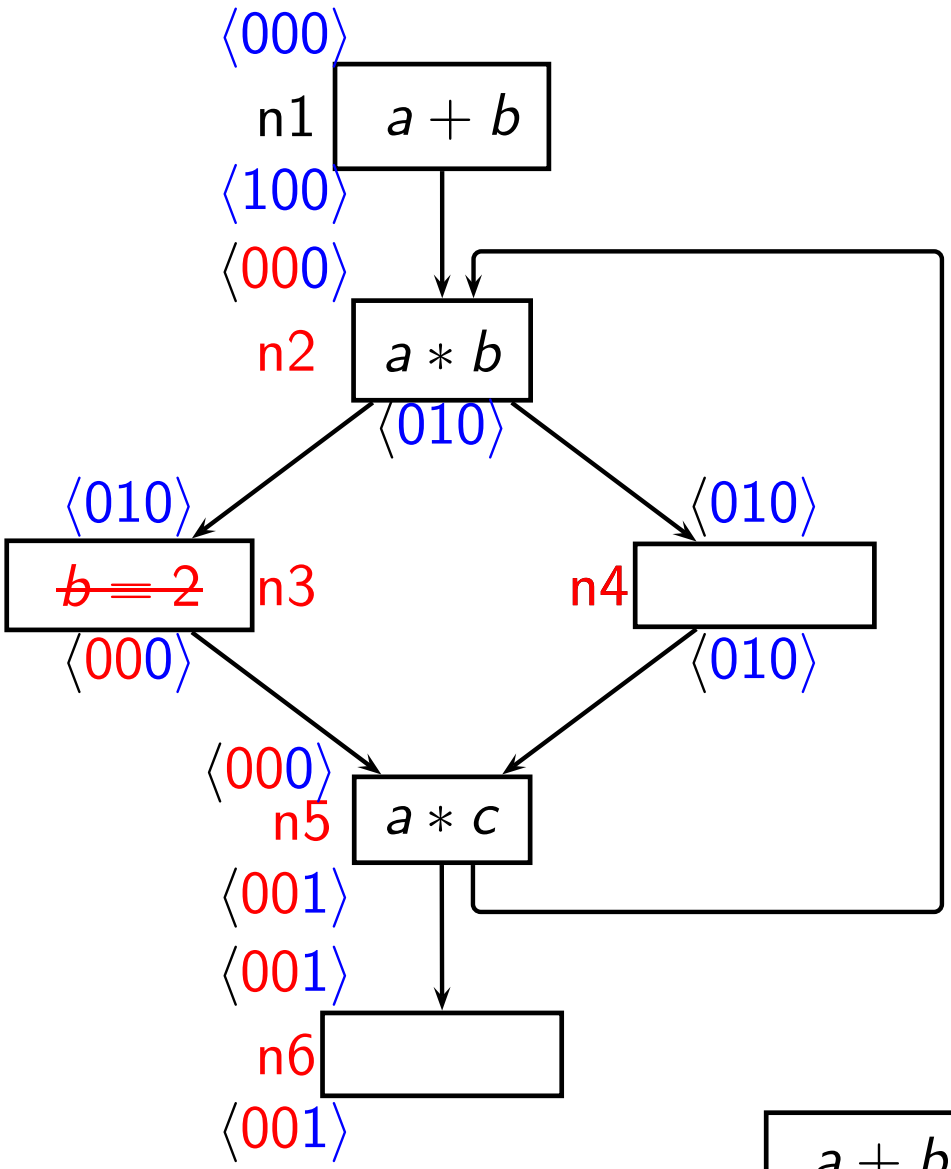
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

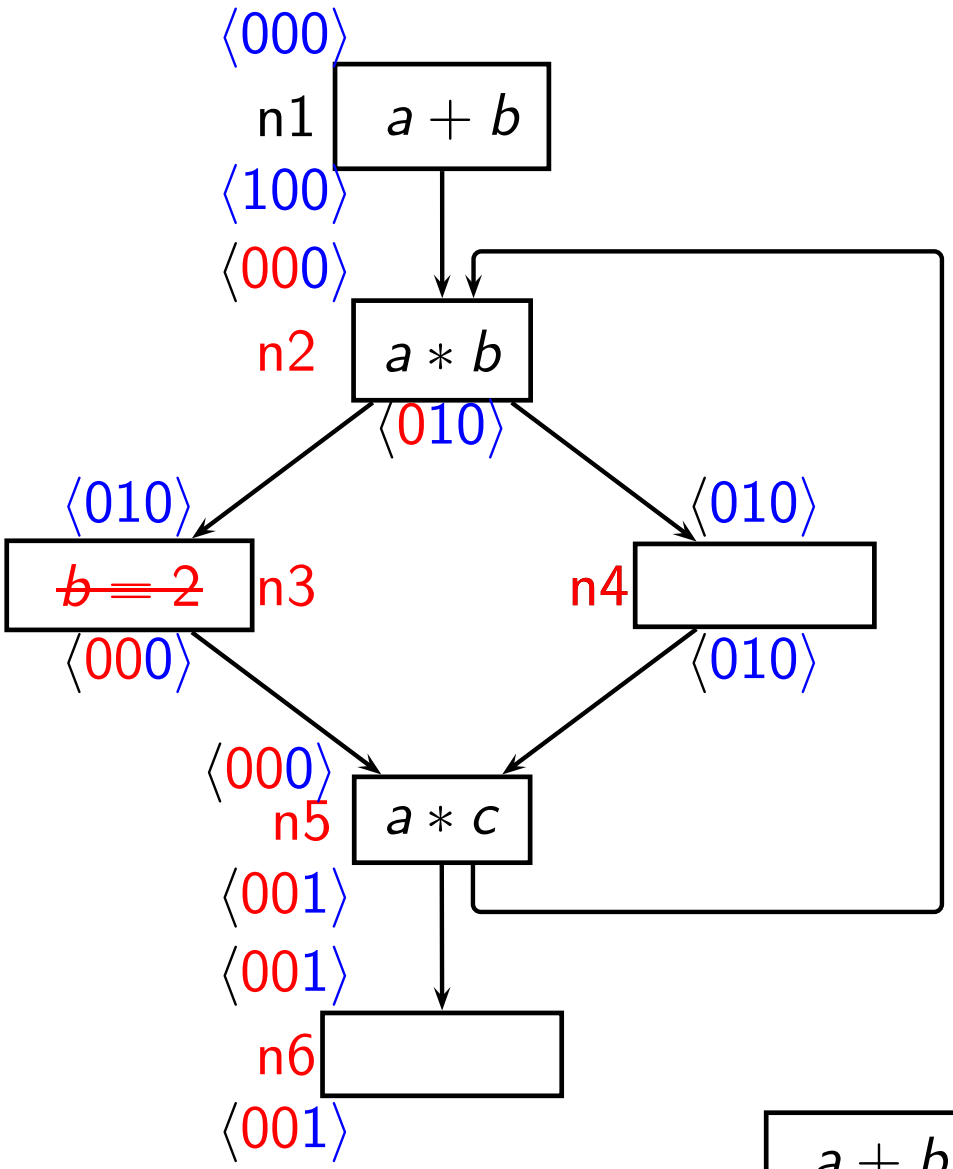
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

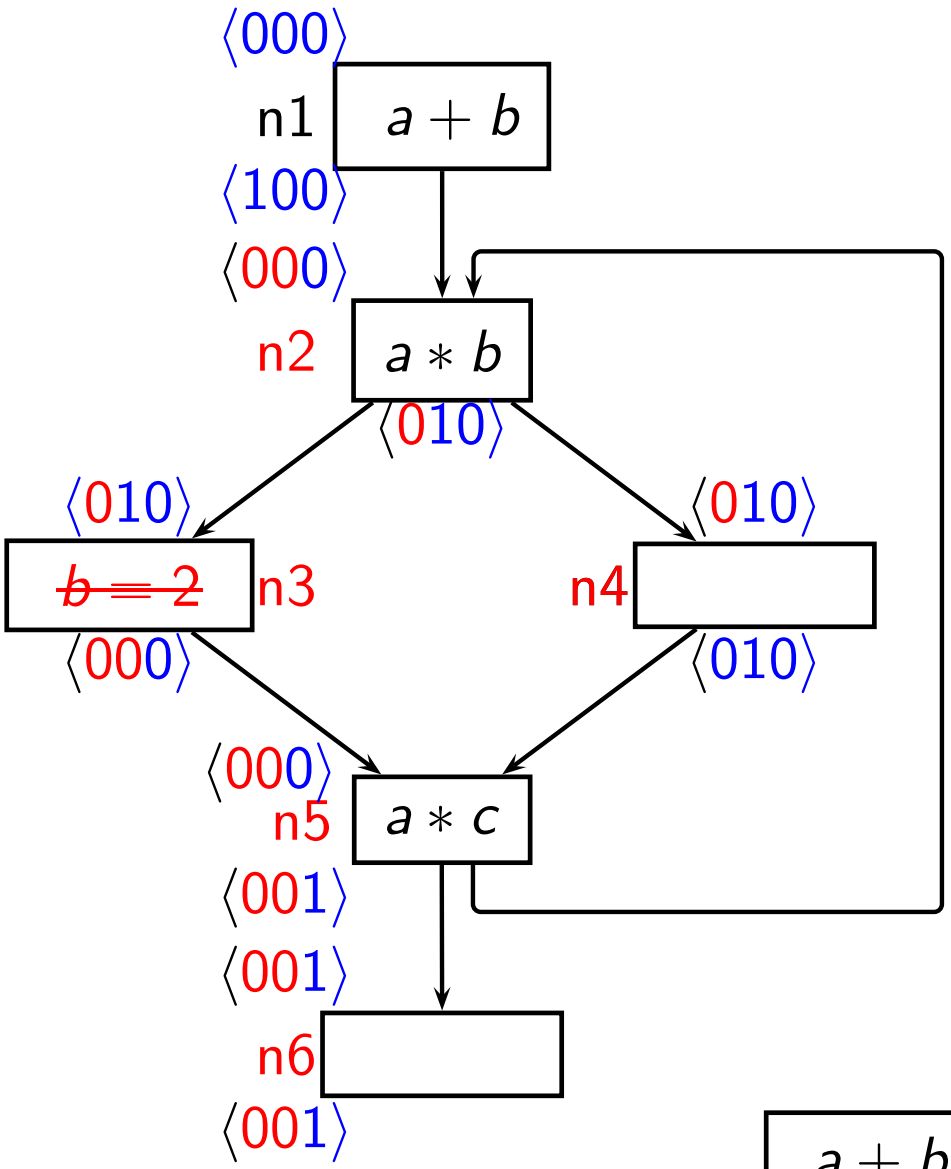
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

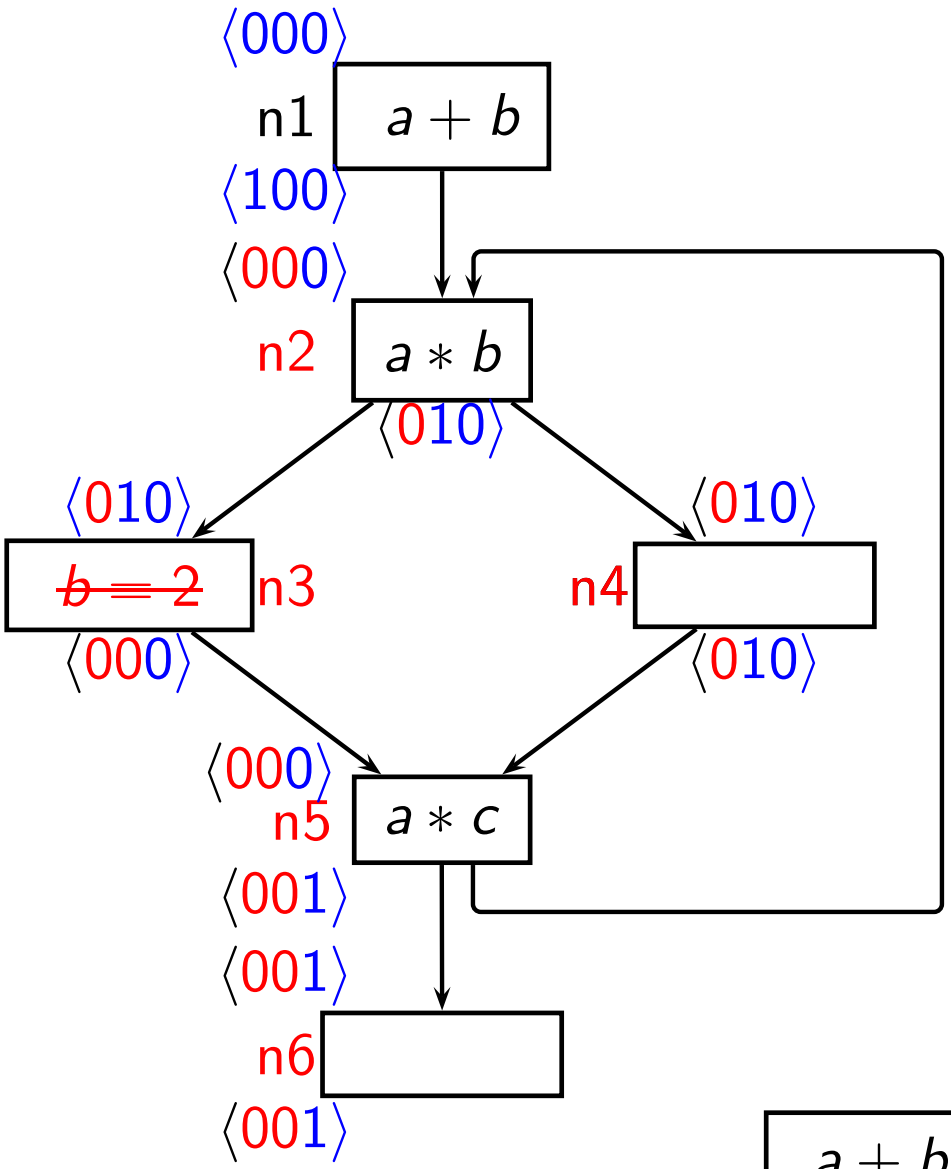
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

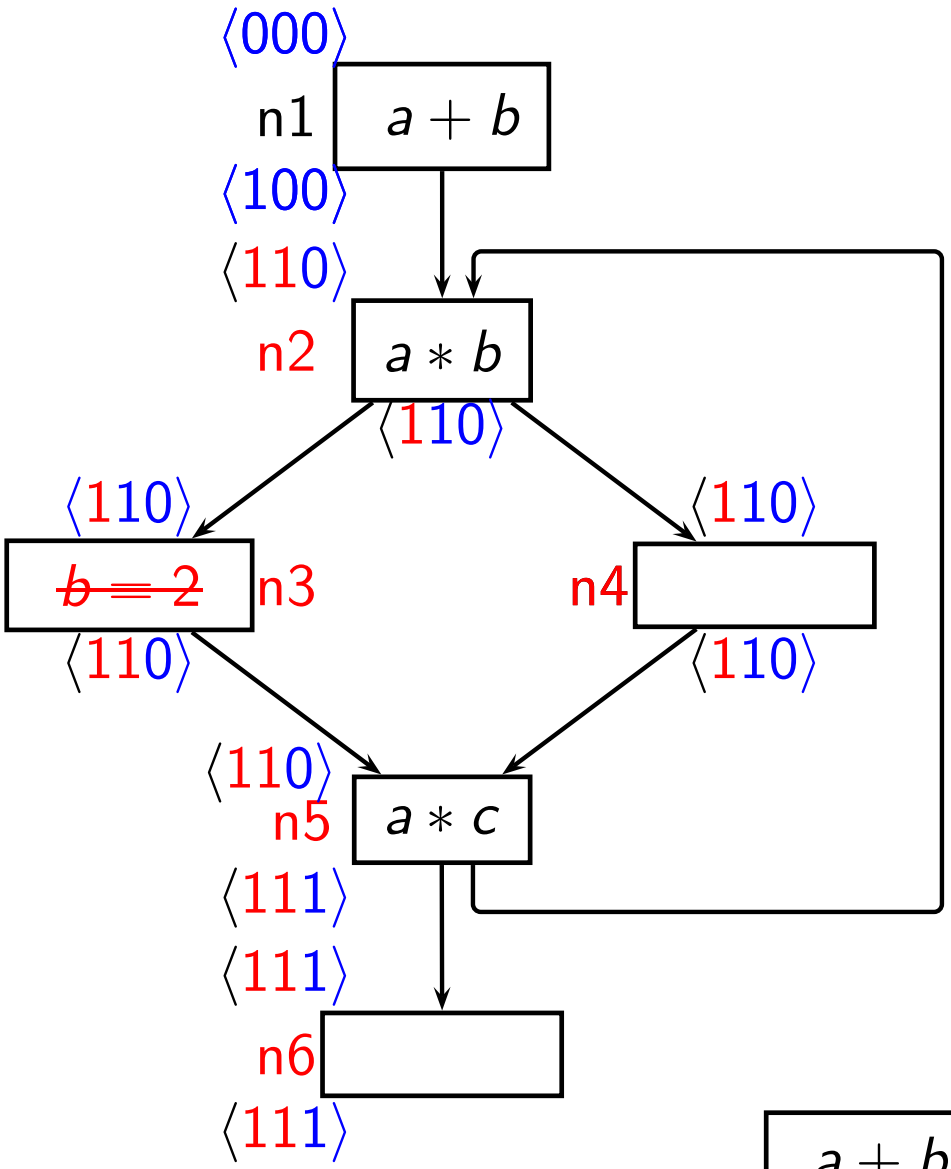
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

Motivating Example - Step 1



Data flow values which may become 1

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.						
2.	1	1	1			
3.	1	1		1		
4.	1	1				
5.	1	1	1	1		
6.	1	1	1	1		

$a+b$	$a*b$	$a*c$

Motivating Example - Step 2

- Find out the data flow values which must remain bottom due to the effect of some other property

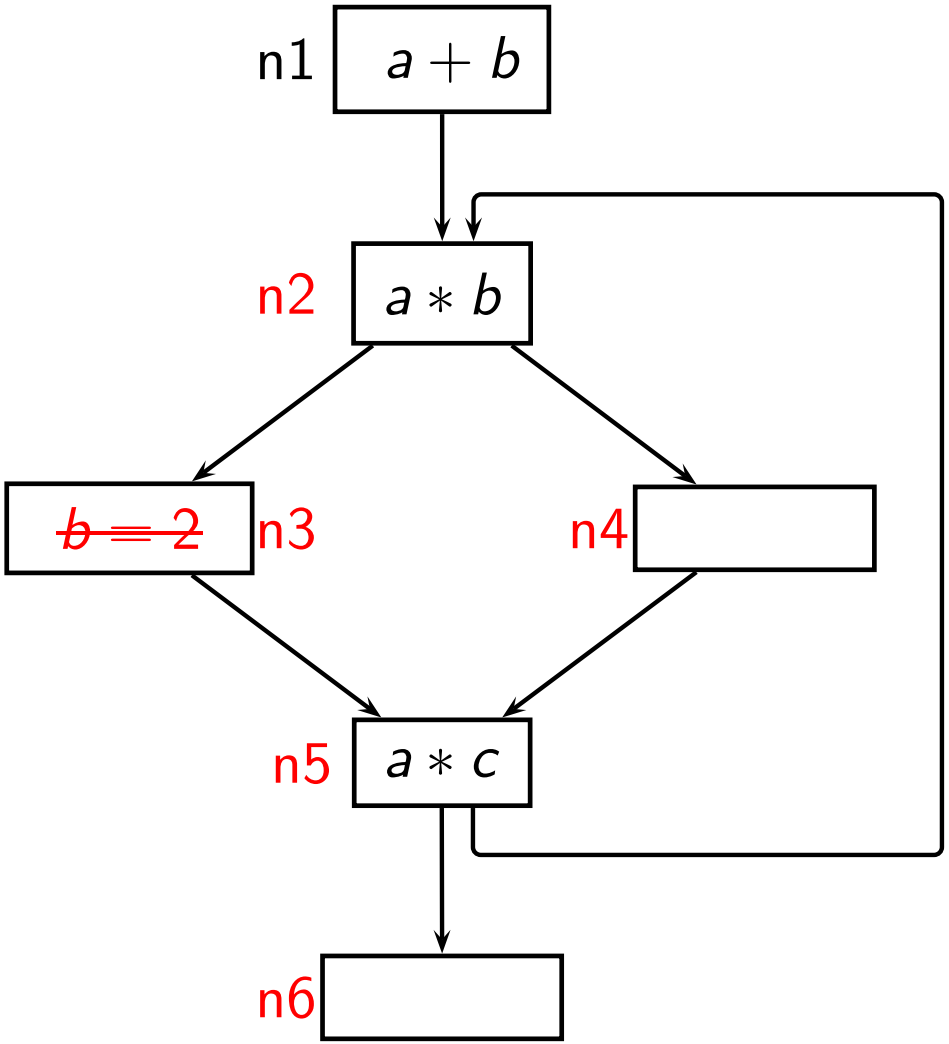
Motivating Example - Step 2

- Find out the data flow values which must remain bottom due to the effect of some other property
  - Identifying Boundary nodes

Motivating Example - Step 2

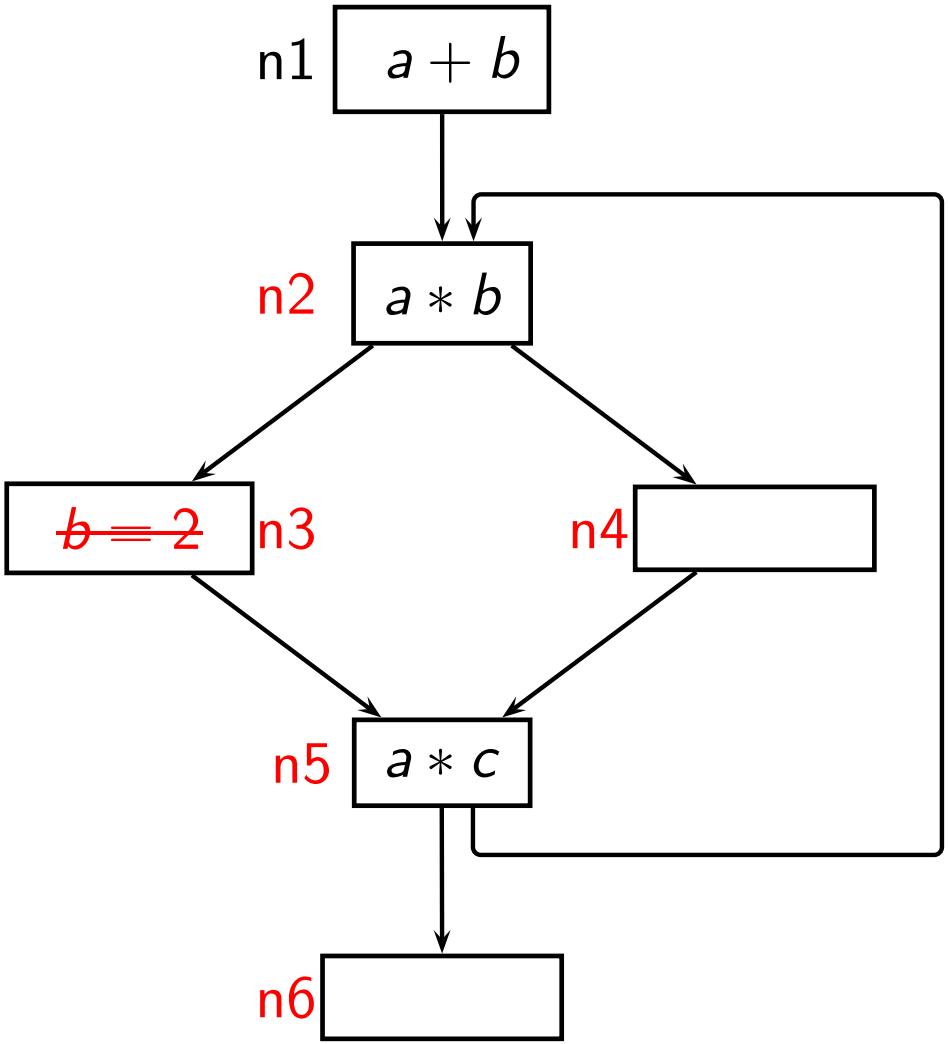
- Find out the data flow values which must remain bottom due to the effect of some other property
  - Identifying Boundary nodes
  - Computing values at boundary nodes and propagating them

Motivating Example - Step 2



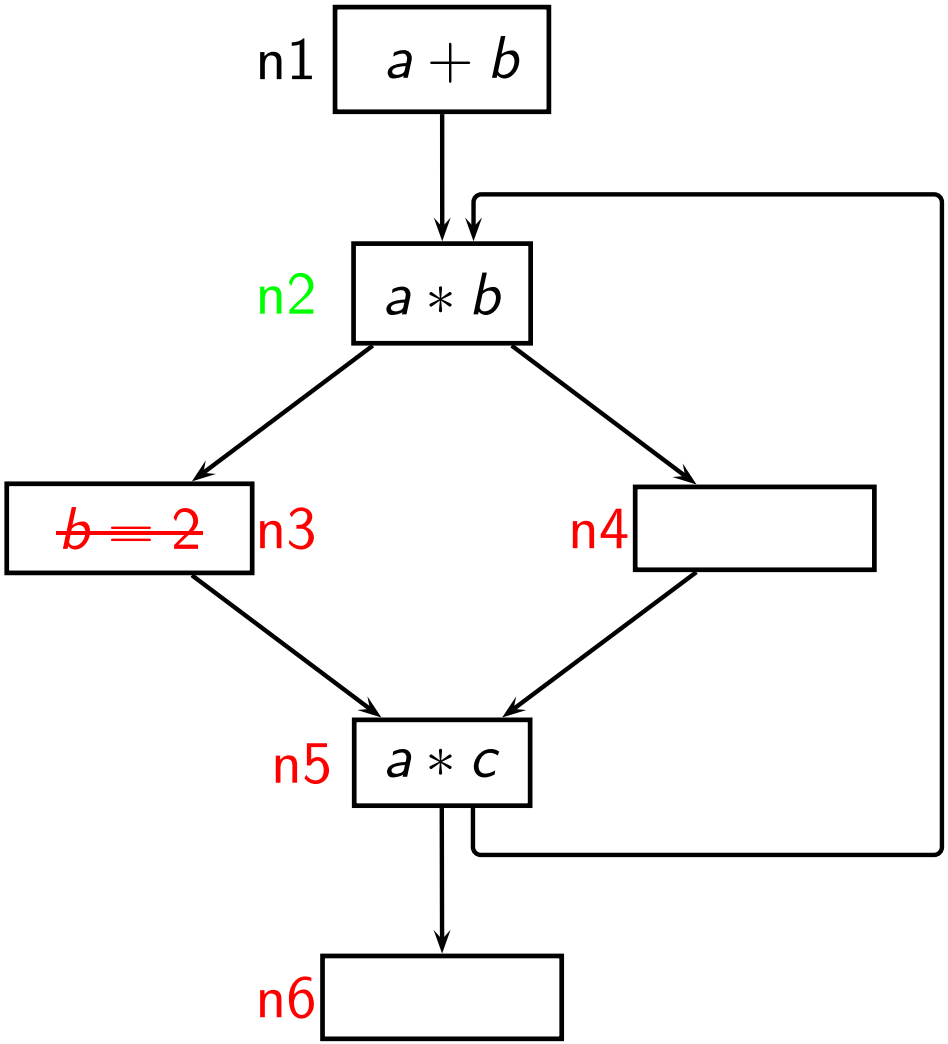
$a + b$	$a * b$	$a * c$

Motivating Example - Step 2



$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

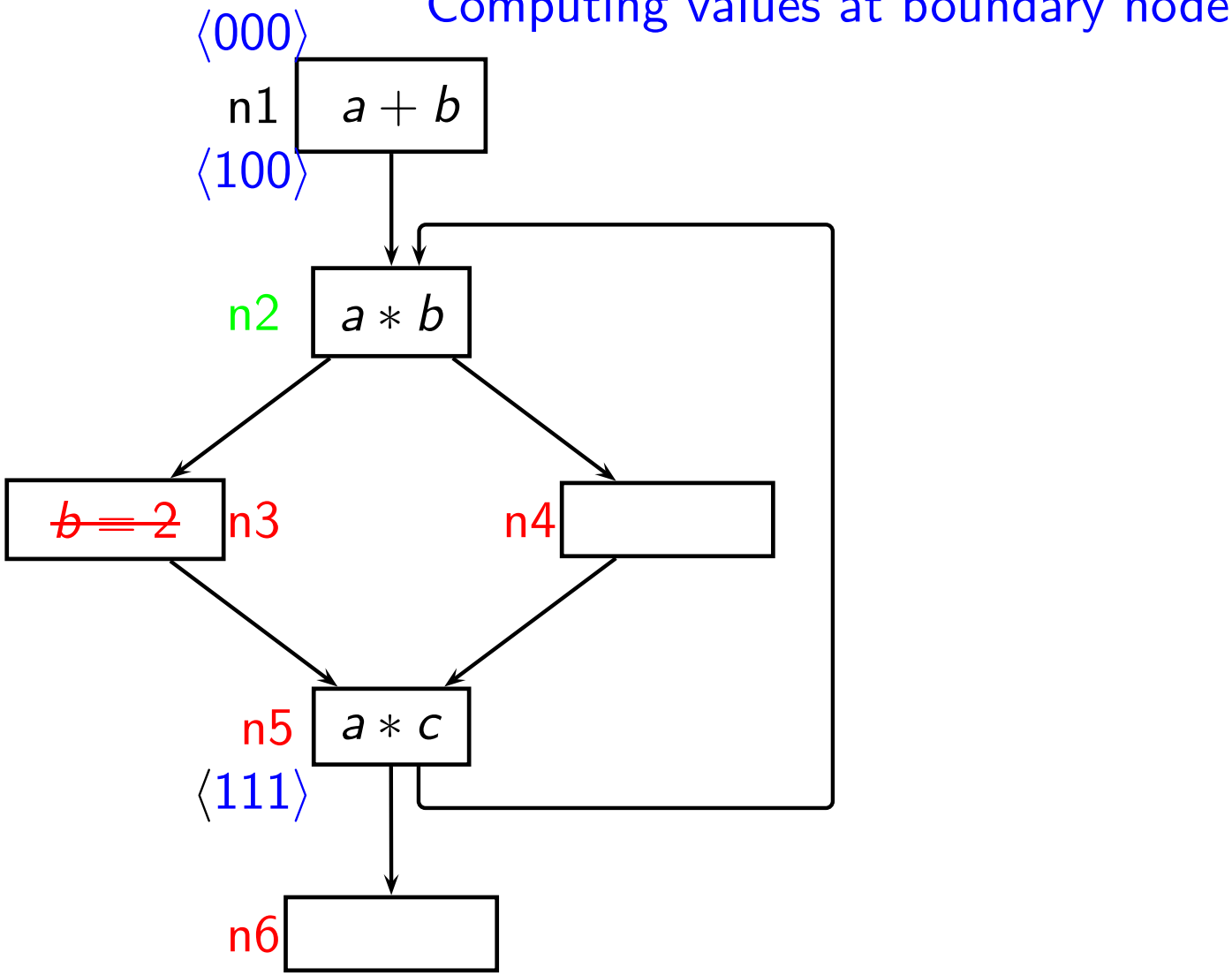


Node 2 is Boundary node

$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

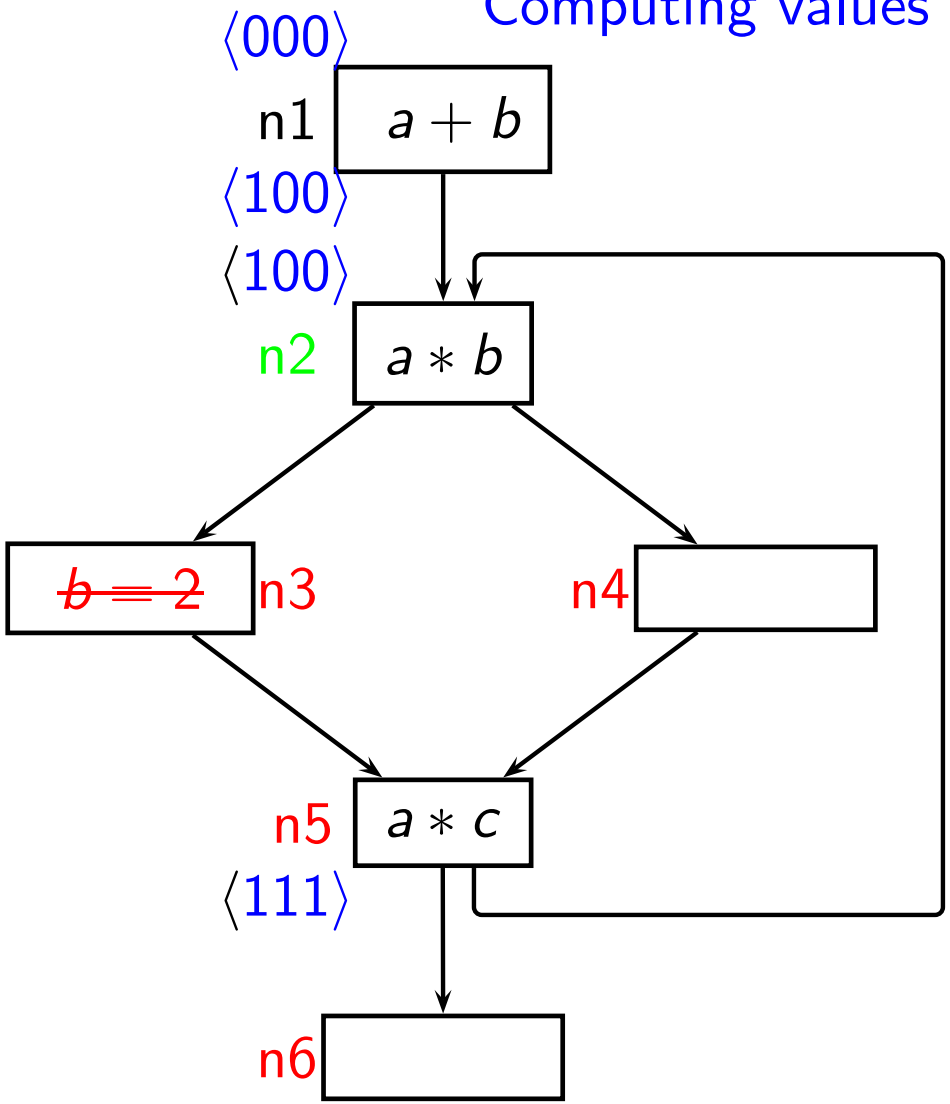
Computing values at boundary node and propagating them



$a+b$	$a*b$	$a*c$

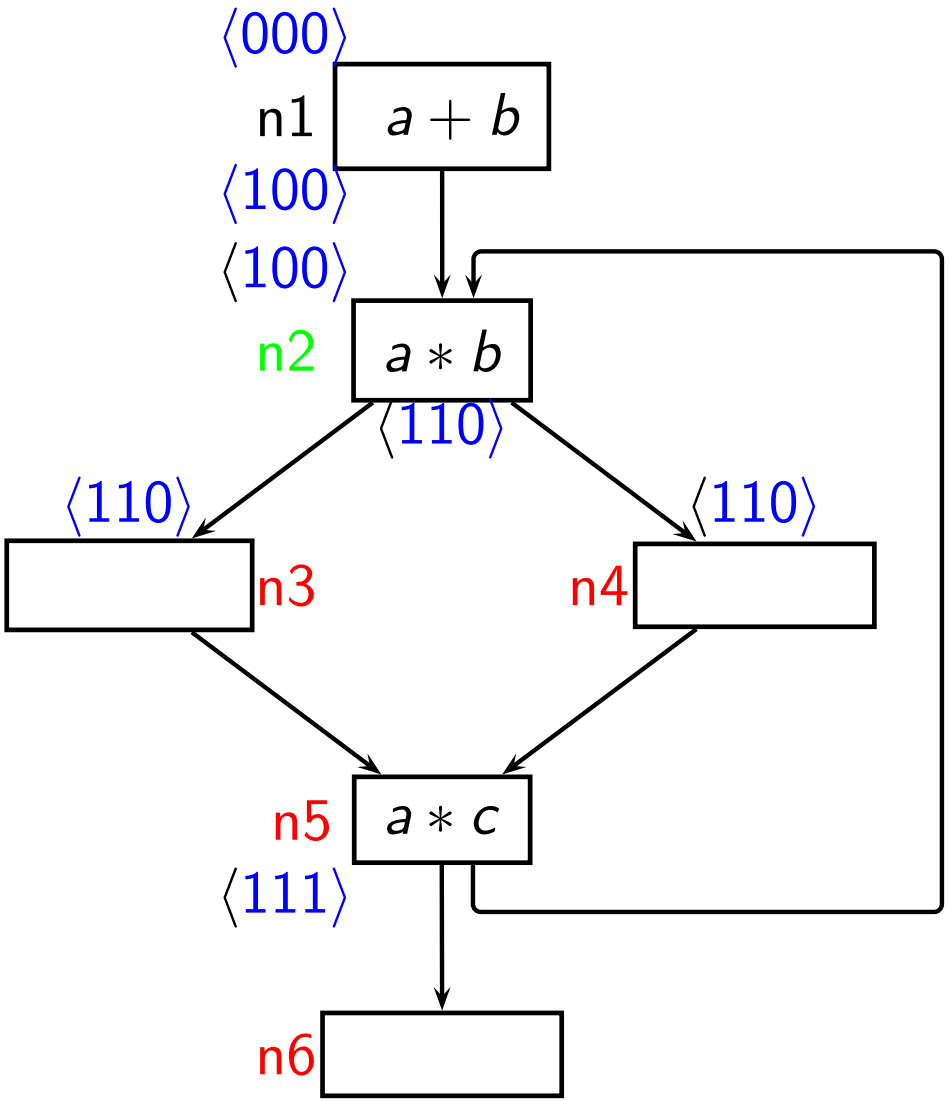
Motivating Example - Step 2

Computing values at boundary node and propagating them



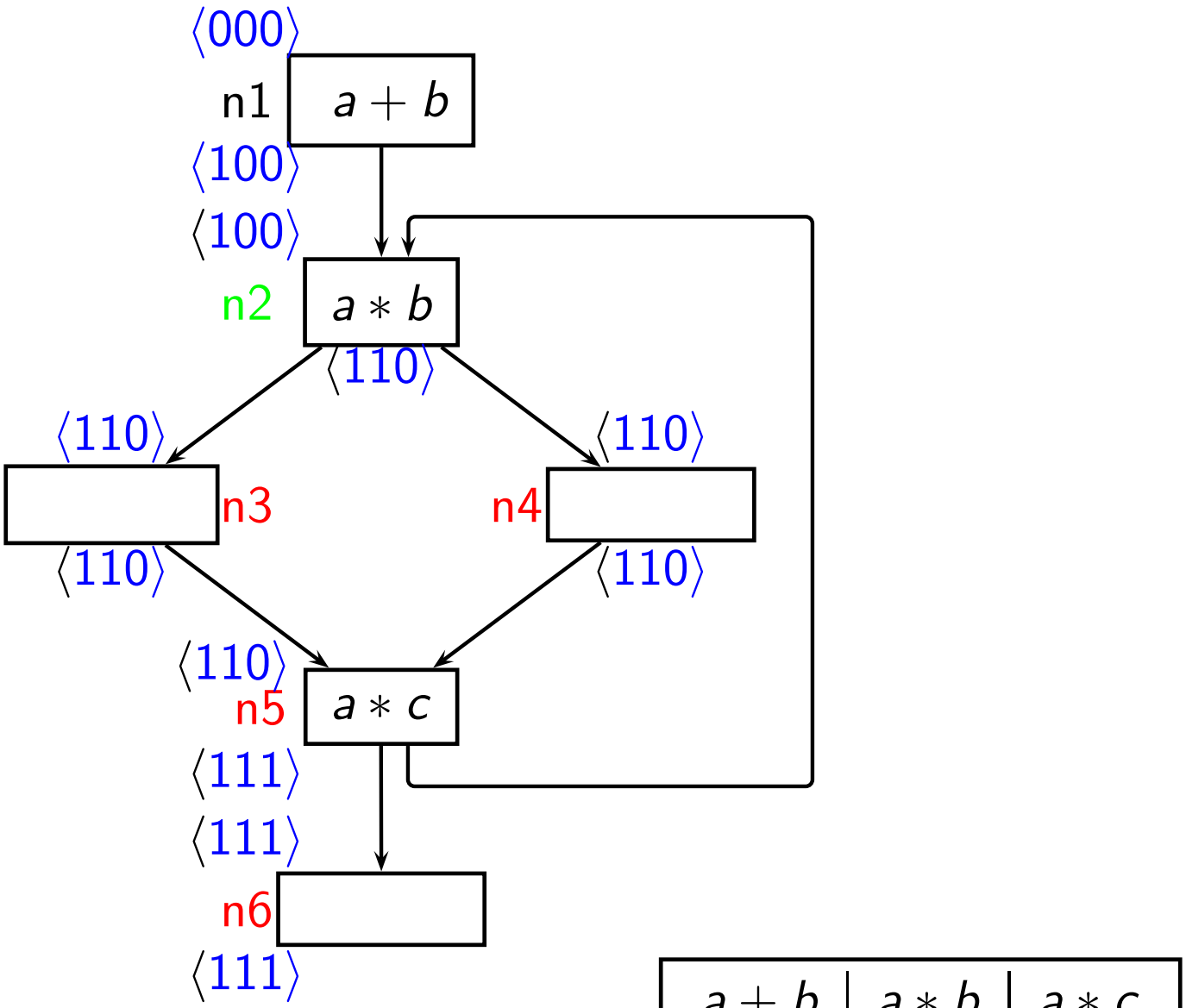
$a+b$	$a*b$	$a*c$

Motivating Example - Step 2



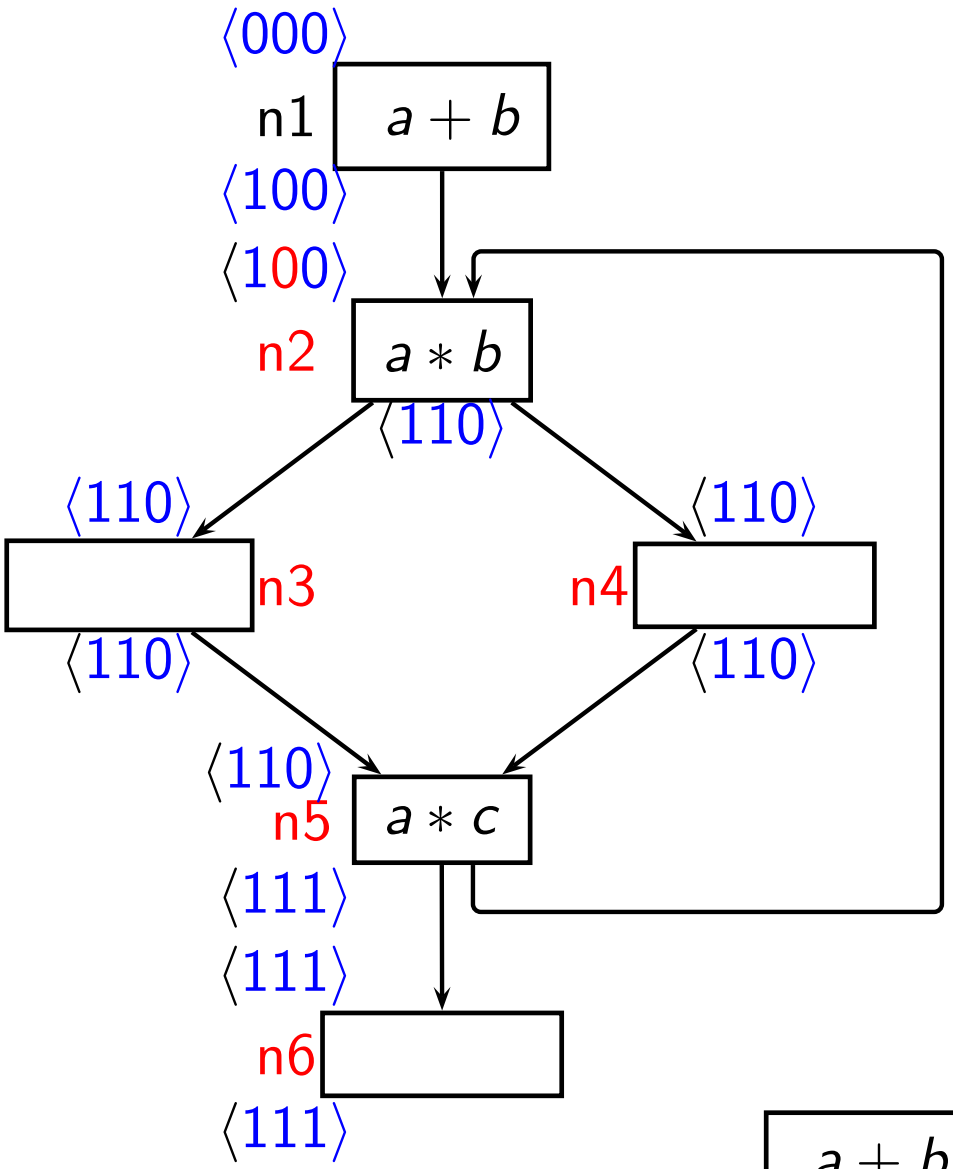
$a + b$	$a * b$	$a * c$

Motivating Example - Step 2



$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

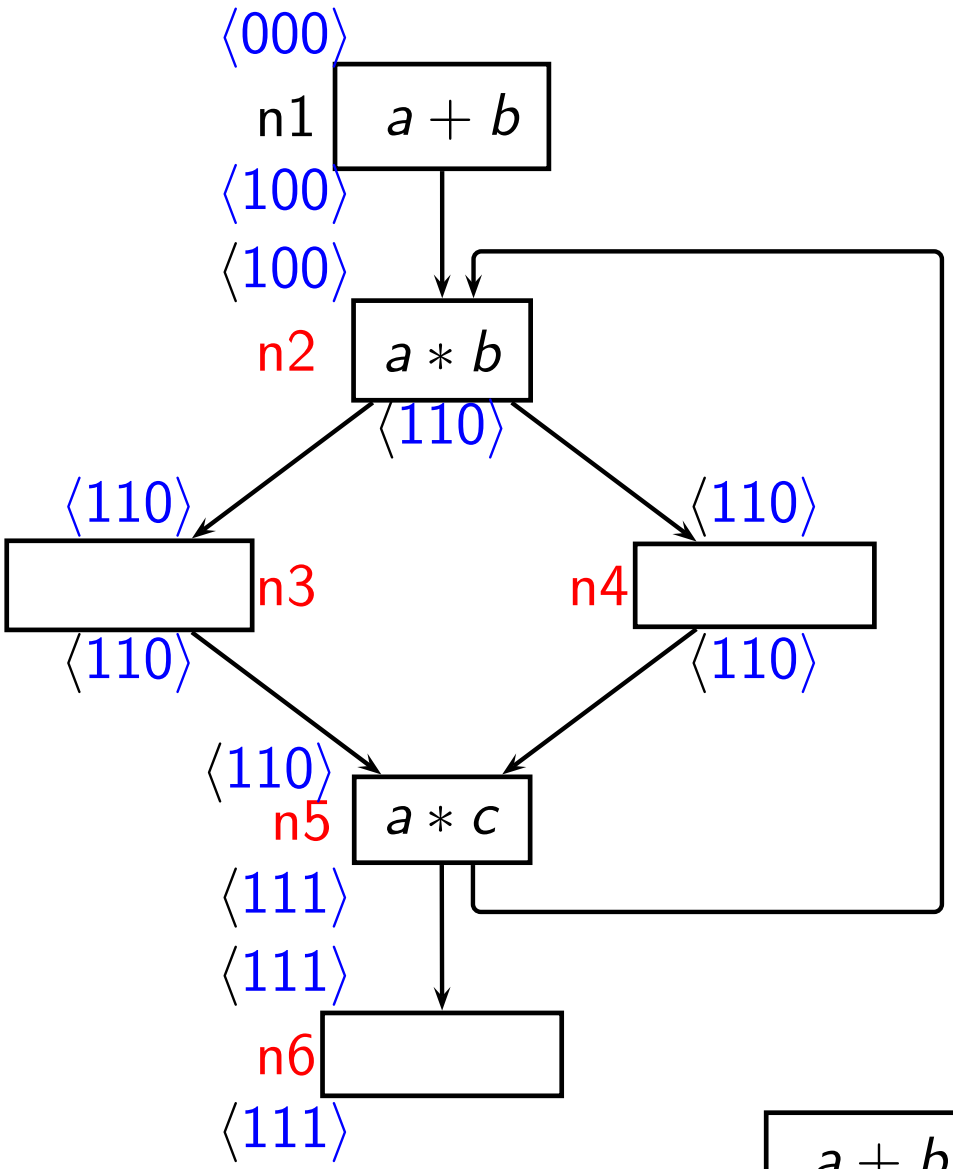


Values which must remain 0

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.						
2.			0			
3.						
4.						
5.						
6.						

$a+b$	$a*b$	$a*c$

Motivating Example - Step 2



Final values

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	1	1	0	1	0	0
3.	1	1	1	1	0	0
4.	1	1	1	1	0	0
5.	1	1	1	1	0	1
6.	1	1	1	1	1	1

$a+b$	$a*b$	$a*c$

Part IV

Thank You !